Approximation of Sweep Surfaces by Tensor Product B-splines*

(Technical Report UUCS-88-008)

Mark Bloomenthal

August 29, 1988
Revised August 24, 1992

Abstract

Tensor product B-spline approximations to surfaces generated by sweeping a (possibly deforming) B-spline cross-section curve along a B-spline axis curve are discussed. A general form for the tensor product B-spline approximation for sweeps is derived and expressed in terms of the approximation of a set of offset curves of the axis curve. The actual algorithm used to generate the approximation depends on the nature of the desired deformation and change in orientation that the cross-section undergoes as it is swept along the axis. Several algorithms for generating tensor product B-spline approximations to sweep surfaces are presented.

*This work was supported in part by DARPA (N00014-88-K-0689). All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

Contents

1 Introduction

A sweep can be defined, in very general terms, as the generation of a curve, surface, or volume by moving a geometric object (a point, curve, surface, or volume) in 3-space. The union of all points in 3-space that are occupied by the object as it is moved, becomes the resulting sweep entity. This idea can be further generalized by allowing the object that is being swept to deform as it is moved.

We are concerned with a restriction of this general sweeping problem: the generation of surfaces by sweeping one 3-space curve (called the **cross-section curve**) along another 3-space curve (called the **axis curve**). This sweep paradigm is attractive in that it reduces the problem of surface design to the simpler problem of curve design. A wide variety of shapes can be described intuitively in terms of a (possibly deformable) cross-section moving along a central axis. Ship hulls, aircraft wings, handles, plumbing fixtures, and gears are a few examples.

In this paper we view the sweep paradigm from within the context of a B-spline based geometric modeling system. We are concerned with tensor product B-spline approximations to surfaces generated by sweeping a B-spline cross-section curve along a B-spline axis curve. The basic approach is that of Coquillart [7] who related the problem of the approximation of sweep surfaces to the problem of the approximation of offset curves.

In Section 3 we give a general parametric form for the sweeps under consideration. We then derive a general form for the tensor product B-spline approximation of sweeps. This approximation is related to the approximation of offset curves of the axis by B-splines. Simple algorithms for sweep and offset approximations are presented in Section 4; they are useful in the case of non-deformable cross-section curves. In Section 5 the simple sweep algorithm is generalized to one for approximating sweeps with deformable cross-sections. Two typical techniques for specifying the cross-section deformation, profile curves and cross-section blending, are discussed. Section 6 deals with the problem of specifying the precise orientation of the cross-section curve as it is swept along the axis curve. Finally in Section 7 we discuss several types of sweeps, each characterized by the method of specifying the cross-section orientation and deformation information.

2 Spline Background

2.1 Univariate B-spline Curves

A univariate B-spline is a series of parametrically defined polynomial pieces joined end to end, with certain smoothness (continuity) constraints at the places where the polynomial pieces are joined. Rational polynomial pieces must be used to allow exact representations for circles and arcs.

B-spline curves are expressed using a particular set of basis functions for the piecewise polynomials known as the B-spline basis. There are many advantages to the use of the B-spline basis, including computational stability and geometric locality (see [13] and [8]).

A rational B-spline of order k over the knot vector $\boldsymbol{\tau}$ is denoted as:

$$\boldsymbol{\gamma}(t) = \frac{\sum_i h_i \mathbf{E}_i B_{i,k,\boldsymbol{\tau}}(t)}{\sum_i h_i B_{i,k,\boldsymbol{\tau}}(t)}.$$

We can consider the rational B-spline $\boldsymbol{\gamma}$ as the projection to $\mathbf{R}^3$ of a polynomial B-spline in $\mathbf{R}^4$ with control points: $\{h_i(\mathbf{E}_i, 1)\}$.

2.2 Nodes of a B-spline Curve

Given a B-spline curve $\boldsymbol{\gamma}$ as above, with knot vector $\boldsymbol{\tau} = (t_1, t_2, \ldots, t_n)$, the ith parametric **node** of $\boldsymbol{\tau}$ is defined as:

$$t_i^* = (t_{i+1} + t_{i+2} + \ldots + t_{i+k-1})/(k-1).$$

The ith node of a curve's knot vector is a first order approximation to the maximum of $B_i(t)$ (see [9]). As such it is an approximation to that point on the curve where the control point $h_i(\mathbf{E}_i, 1)$ exerts its maximal influence.

2.3 Tensor Product B-spline Surfaces

The tensor product B-spline formulation is simply a method of combining univariate B-spline curves to produce functions of two variables, thereby reducing the problem of surface representation to the simpler problem of the representation of curves.

A rational tensor product B-spline surface is denoted as:

$$\boldsymbol{\sigma}(u,v) = \frac{\sum_i \sum_j h_i w_j \mathbf{E}_{ij} B_j(u) N_i(v)}{\sum_i \sum_j h_i w_j B_j(u) N_i(v)}.$$

2.4 B-spline Curve Refinement

Given any B-spline curve with a particular knot vector, $\boldsymbol{\tau}$, the same space curve can be represented using a different knot vector, $\boldsymbol{\tau}'$, if $\boldsymbol{\tau}'$ is a refined partition of $\boldsymbol{\tau}$. The process by which a B-spline curve is represented over a refined knot vector is known as **B-spline refinement.**

B-spline refinement has the very important property of convergence of the control polygon to the curve. As more values are added to a knot vector and a curve refined, the control polygon for the curve approaches the actual curve in the parametric area where the new knot vector values are added. If a curve is highly refined over its entire parametric domain, the control polygon can be made arbitrarily close to the curve everywhere.

There exist very efficient and elegant algorithms for the refinement of B-spline curves [3]. Points on B-spline curves and surfaces can be evaluated by using special cases of these algorithms.

3 Framework

3.1 General Parametric Form for a Sweep Surface

A general parametric form for the sweeps under consideration is:

$$\boldsymbol{\sigma}(u,v) = \mathbf{a}(u) + \boldsymbol{\mathcal{F}}(u)\mathbf{c}(v), \tag{1}$$

where:

$\mathbf{c}(v)$ is a parametrically defined general 3-D curve called the **cross-section curve,**

$\mathbf{a}(u)$ is a parametrically defined general 3-D curve, called the **axis curve,** along which the cross-section curve is swept, and

$\mathcal{F}(u) = \left[\begin{array}{ccc} \mathbf{X}(u) & \mathbf{Y}(u) & \mathbf{Z}(u) \end{array} \right]$ will be referred to as a **local coordinate frame** or **local orientation frame.** $\mathbf{X}(u)$, $\mathbf{Y}(u)$, and $\mathbf{Z}(u)$ are vector valued functions that together specify three orthonormal vectors at each value of the parameter u in the domain of $\mathbf{a}(u)$. (In equation (1) each of these functions is taken as a column vector.) These functions provide orientation information along the axis curve.

Equation (1) specifies rigid body motion of the cross-section curve along the axis curve. At a given value of u, (1) specifies an affine transformation of the cross-section curve, with translation given by $\mathbf{a}(u)$ and rotation given by $\mathcal{F}(u)$ (see Figures 1 and 2).

Equation (1) includes many cases unwanted in a design context such as self-intersecting, degenerate, or discontinuous surfaces. The conditions under which such cases arise are not discussed in this paper. It is understood however that we desire our sweep surfaces to be at least continuous. Thus we assume the axis curve $\mathbf{a}(u)$, the cross-section curve $\mathbf{c}(v)$, and orientation frame $\mathcal{F}(u)$ to be continuous.

We can extend the notion of a sweep surface as given in (1) by allowing the cross-section curve to deform as it is swept along the axis curve. That is, we let the cross-section curve be a function of u as well, yielding:

$$\boldsymbol{\sigma}(u, v) = \mathbf{a}(u) + \mathcal{F}(u)\mathbf{c}(u, v).$$

In the next section we introduce the concept of an offset curve. Offset curves are used in Section 3.4 to formulate approximations of sweep surfaces by tensor product B-splines.

3.2 Offset Curve Definition

Isolines in u of $\boldsymbol{\sigma}(u, v)$ are, as indicated above, simply affine transformations of the cross-section. What are isolines in v of $\boldsymbol{\sigma}(u, v)$?

Isolines in v can be considered sweeps of cross-sections consisting of a single point. For a given value of $v = v'$, let $\mathbf{c}(v')$ be the point $\mathbf{P}$. Then:

$$\boldsymbol{\sigma}_{v'}(u) = \mathbf{a}(u) + \mathcal{F}(u)\mathbf{P}.$$

We can interpet this expression as follows: We generate $\boldsymbol{\sigma}_{v'}(u)$ by offsetting each point of $\mathbf{a}(u)$ by the *constant* vector $\mathbf{P}$ relative to the *local*

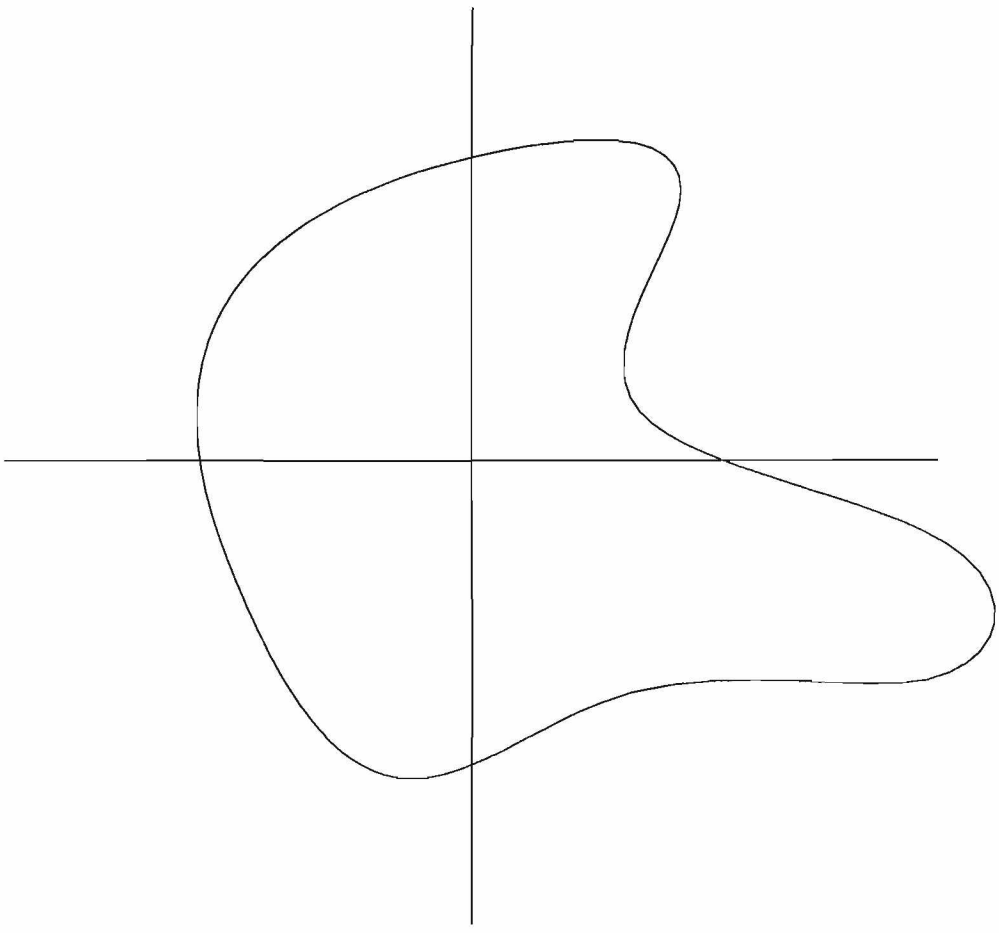

Figure 1: A cross-section curve, $\mathbf{c}(v)$, shown here in the xy plane.

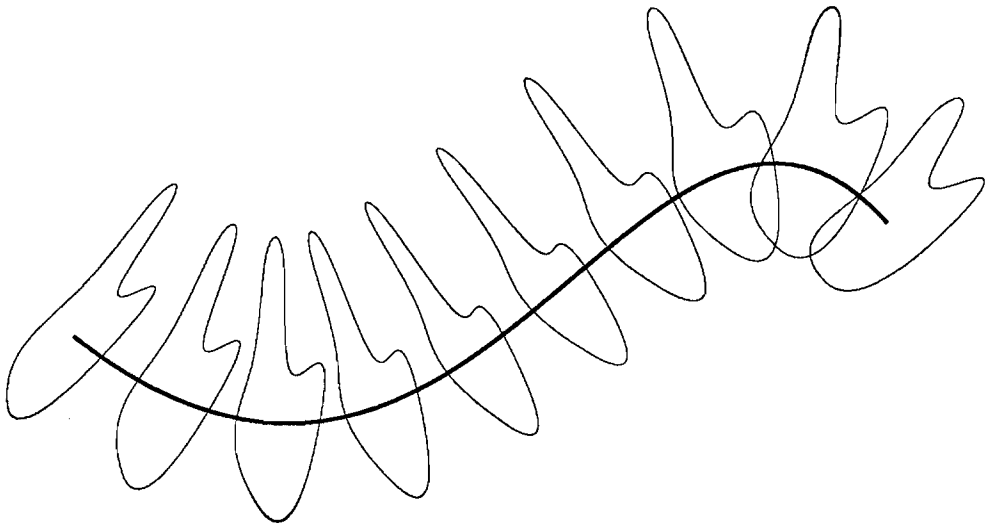

Figure 2: Sweep of cross-section curve $\mathbf{c}(v)$ along axis curve $\mathbf{a}(u)$.

coordinate frame $\mathcal{F}(u)$. For this reason, we call $\boldsymbol{\sigma}_{v'}(u)$ an **offset curve** of $\mathbf{a}(u)$ (see Figures 3 and 4).

For $\mathbf{V} = (x, y, z)$, we define:

$$\mathbf{o}(t) = \boldsymbol{\gamma}(t) + \mathcal{F}(t) \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2}$$

to be the offset curve of $\boldsymbol{\gamma}$ by the vector $\mathbf{V}$ relative to the frame $\mathcal{F}(t)$.

This definition is a simple generalization of the more usual definition of an offset curve. For $\mathcal{F}(t) = \begin{bmatrix} \mathbf{N}(t) & \mathbf{B}(t) & \mathbf{T}(t) \end{bmatrix}$, the **Frenet frame**, consisting of the **principal normal**, the **binormal**, and the **unit tangent** functions for $\boldsymbol{\gamma}(t)$, we have:

$$\mathbf{o}(t) = \boldsymbol{\gamma}(t) + \begin{bmatrix} \mathbf{N}(t) & \mathbf{B}(t) & \mathbf{T}(t) \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

For $\mathbf{V} = (d, 0, 0)$ we have:

$$\mathbf{o}(t) = \boldsymbol{\gamma}(t) + d\mathbf{N}(t),$$

which is the usual definition for the offset of a planar curve by the signed distance d (although this definition ignores possible degeneracies in the Frenet frame and self intersecting results).

3.3 Sweeps and Offsets of B-spline Curves

We are interested in tensor product B-spline representations for surfaces generated by sweeping a B-spline cross-section curve along a B-spline axis curve in accordance with (1). Whether the resulting expression is exactly representable by a rational tensor product B-spline depends on our choice for the frame $\boldsymbol{\mathcal{F}}(u)$.

In Section 6 we discuss the issue of selecting the frame $\boldsymbol{\mathcal{F}}(u)$ of equation (1). In general, our choices for the functions $\mathbf{X}(u), \mathbf{Y}(u)$, and $\mathbf{Z}(u)$ are not representable exactly as rational B-splines. As such, sweeps of B-spline cross-sections along B-spline axis curves are not, in general, exactly representable as tensor product B-splines. Similarly, offsets of B-spline curves as given by (2) are not, in general, exactly representable as B-splines.

3.4 Tensor Product B-spline Approximation

We approximate $\boldsymbol{\sigma}(u, v)$ of equation (1) by a rational tensor product B-spline, where we assume that both $\mathbf{a}(u)$ and $\mathbf{c}(v)$ are given as rational B-spline curves. We make use of the tensor product B-spline form to reduce the problem of approximating a sweep surface to that of approximating a set of offset curves.

For axis curve $\mathbf{a}(u)$, cross-section curve $\mathbf{c}(v)$, and frame $\boldsymbol{\mathcal{F}}(u)$ we consider the sweep surface given by:

$$\begin{aligned} \boldsymbol{\sigma}(u, v) &= \mathbf{a}(u) + \boldsymbol{\mathcal{F}}(u)\mathbf{c}(v) \\ &= \mathcal{T}(u)(\mathbf{c}(v)), \end{aligned}$$

where $\mathcal{T}(u)$ is an affine transformation.

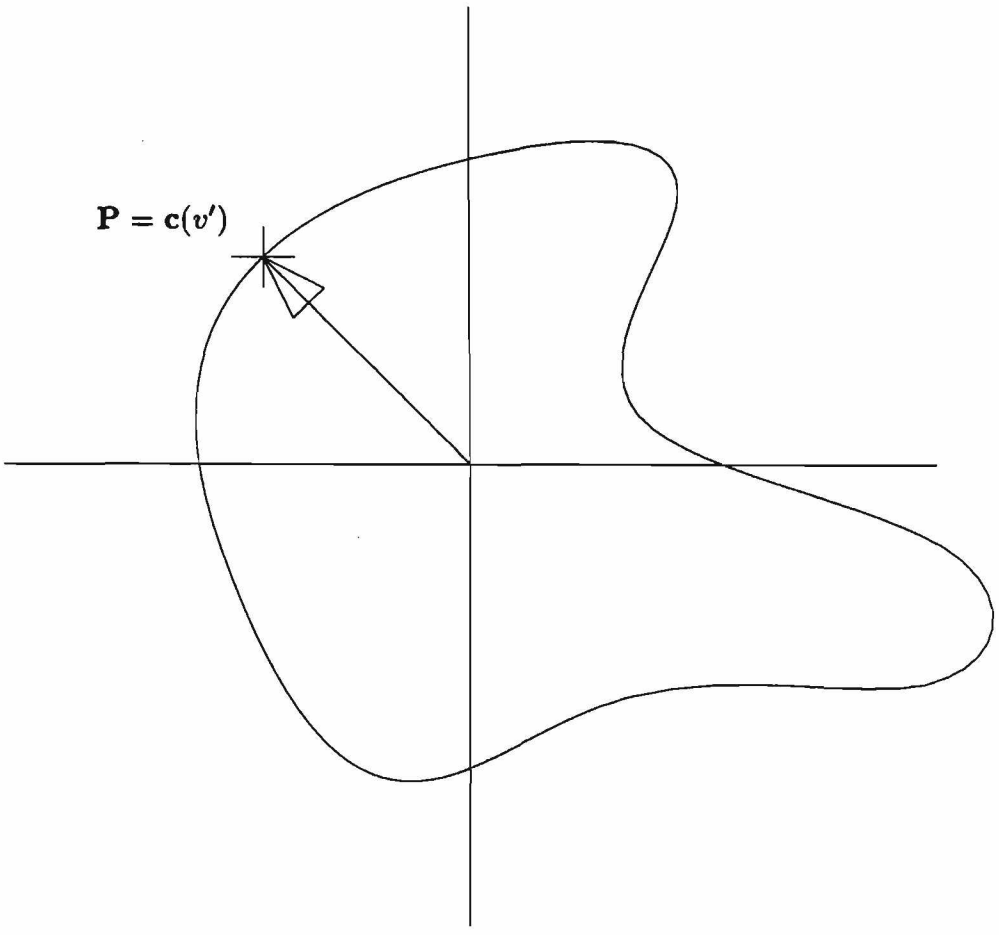

Figure 3: Offset vector $\mathbf{P}$ generated on cross-section $\mathbf{c}(v)$.

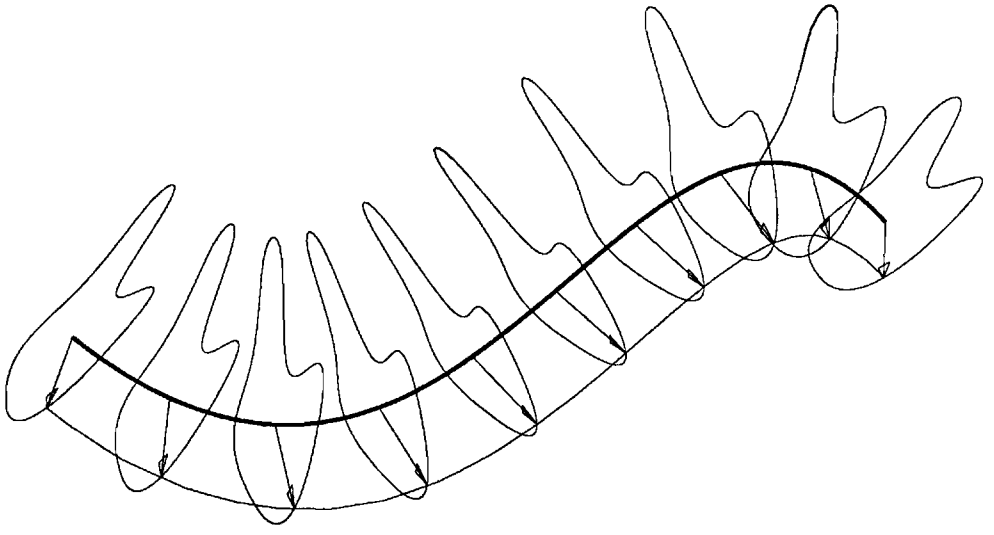

Figure 4: Isoline $\boldsymbol{\sigma}_{v'}(u)$ of the sweep surface $\boldsymbol{\sigma}(u,v)$ can be considered an offset of the axis curve by the *constant* vector $\mathbf{P} = \mathbf{c}(v')$ relative to a *local* orientation frame on the axis.

With $\mathbf{c}(v)$ a rational B-spline curve:

$$\begin{aligned}
\boldsymbol{\sigma}(u,v) &= \mathcal{T}(u)\left(\frac{\sum_i h_i^c \mathbf{C}_i B_i^c(v)}{\sum_i h_i^c B_i^c(v)}\right) \\
&= \frac{\sum_i h_i^c \mathcal{T}(u)(\mathbf{C}_i) B_i^c(v)}{\sum_i h_i^c B_i^c(v)}, \qquad \text{since } \mathcal{T}(u) \text{ is affine (see Appendix A).} \\
&= \frac{\sum_i h_i^c \boldsymbol{\alpha}_i(u) B_i^c(v)}{\sum_i h_i^c B_i^c(v)} \qquad (3)
\end{aligned}$$

where:

$\{\mathbf{C}_i\}_i$ are the Euclidian projections of the B-spline control points for $\mathbf{c}(v)$,
$\{h_i^c\}_i$ are the homogeneous coordinates of the control points for $\mathbf{c}(v)$,
$\{B_i^c\}_i$ are the B-spline basis functions for $\mathbf{c}(v)$, and

$$\boldsymbol{\alpha}_i(u) = \mathcal{T}(u)(\mathbf{C}_i) = \mathbf{a}(u) + \mathcal{F}(u)\mathbf{C}_i \qquad (4)$$

is the offset curve of $\mathbf{a}(u)$ by the vector from the origin to $\mathbf{C}_i$ relative to the frame $\mathcal{F}(u)$.

Our end goal is to approximate $\boldsymbol{\sigma}(u,v)$ by a tensor product B-spline. Consider,

$$\hat{\boldsymbol{\sigma}}(u,v) = \frac{\sum_i h_i^c \hat{\boldsymbol{\alpha}}_i(u) B_i^c(v)}{\sum_i h_i^c B_i^c(v)}, \tag{5}$$

where the $\hat{\boldsymbol{\alpha}}_i(u)$ are rational B-spline approximations for the offset curves $\boldsymbol{\alpha}_i(u)$. What conditions on the $\hat{\boldsymbol{\alpha}}_i(u)$ must hold in order that $\hat{\boldsymbol{\sigma}}(u,v)$ be representable as a tensor product B-spline surface?

As a tensor product B-spline, $\hat{\boldsymbol{\sigma}}(u,v)$ must contain a rectangular array of control points. This implies that the $\hat{\boldsymbol{\alpha}}_i(u)$ must all have the same number of control points. Further, only two sets of basis functions occur in the tensor product form which implies that the $\hat{\boldsymbol{\alpha}}_i(u)$ must share the same set of B-spline basis functions. Thus:

$$\hat{\boldsymbol{\alpha}}_i(u) = \frac{\sum_j h_{ij}^\alpha \mathbf{E}_{ij} B_j^\alpha(u)}{\sum_j h_{ij}^\alpha B_j^\alpha(u)},$$

where:

$\{\mathbf{E}_{ij}\}_j$ are the Euclidian projections of the B-spline control points for $\hat{\boldsymbol{\alpha}}_i(u)$, $\{h_{ij}^\alpha\}_j$ are the homogeneous coordinates of the control points for $\hat{\boldsymbol{\alpha}}_i(u)$, and $\{B_j^\alpha\}_j$ are the B-spline basis functions for $\hat{\boldsymbol{\alpha}}_i(u)$.

We now write:

$$\hat{\boldsymbol{\sigma}}(u,v) = \frac{\sum_i h_i^c \left[\left(\sum_j h_{ij}^\alpha \mathbf{E}_{ij} B_j^\alpha(u)\right) / \left(\sum_j h_{ij}^\alpha B_j^\alpha(u)\right)\right] B_i^c(v)}{\sum_i h_i^c B_i^c(v)}.$$

This last result reduces to a tensor product B-spline under the additional assumption that the control polygons for the $\hat{\boldsymbol{\alpha}}_i(u)$ all share the same set of homogeneous coordinates. That is:

$$\hat{\boldsymbol{\alpha}}_i(u) = \frac{\sum_j h_j^\alpha \mathbf{E}_{ij} B_j^\alpha(u)}{\sum_j h_j^\alpha B_j^\alpha(u)},$$

yielding,

$$\begin{aligned}\hat{\boldsymbol{\sigma}}(u,v) &= \frac{\sum_i h_i^c \left[\left(\sum_j h_j^\alpha \mathbf{E}_{ij} B_j^\alpha(u)\right) / \left(\sum_j h_j^\alpha B_j^\alpha(u)\right)\right] B_i^c(v)}{\sum_i h_i^c B_i^c(v)} \\ &= \frac{\sum_i \sum_j h_{ij} \mathbf{E}_{ij} B_j^\alpha(u) B_i^c(v)}{\sum_i \sum_j h_{ij} B_j^\alpha(u) B_i^c(v)}, \end{aligned} \tag{6}$$

where $h_{ij} = h_j^\alpha h_i^c$.

Since the $B_i^c(v)$ are a basis, we note that:

$$\hat{\boldsymbol{\sigma}}(u,v) = \boldsymbol{\sigma}(u,v) \text{ iff } \hat{\boldsymbol{\alpha}}_i(u) = \boldsymbol{\alpha}_i(u) \text{ for all } i. \tag{7}$$

In general this is not the case since, as stated in Section 3.3, the offset of a rational B-spline curve is not, in general, exactly representable as a rational B-spline.

However, an error bound for the $\hat{\boldsymbol{\alpha}}_i(u)$ also provides an error bound for $\hat{\boldsymbol{\sigma}}(u,v)$, that is,

$$\|\boldsymbol{\alpha}_i(u) - \hat{\boldsymbol{\alpha}}_i(u)\| \leq \epsilon \text{ for all } i \;\Rightarrow \|\boldsymbol{\sigma}(u,v) - \hat{\boldsymbol{\sigma}}(u,v)\| \leq \epsilon. \tag{8}$$

Since $h_i^c \geq 0$, $B_i^c(v) \geq 0$ for all i and v, then from (3) and (5),

$$\begin{aligned}\|\boldsymbol{\sigma}(u,v) - \hat{\boldsymbol{\sigma}}(u,v)\| &= \frac{\|\sum_i h_i^c(\boldsymbol{\alpha}_i(u) - \hat{\boldsymbol{\alpha}}_i(u)) B_i^c(v)\|}{\sum_i h_i^c B_i^c(v)} \\ &\leq \frac{\sum_i h_i^c \|\boldsymbol{\alpha}_i(u) - \hat{\boldsymbol{\alpha}}_i(u)\| B_i^c(v)}{\sum_i h_i^c B_i^c(v)} \leq \frac{\sum_i h_i^c \epsilon B_i^c(v)}{\sum_i h_i^c B_i^c(v)} = \epsilon. \end{aligned}$$

Note that since $\sum_i B_i^c(v) \equiv 1$, if $h_i^c = 1$ for all i, then the equation is for polynomial splines and the same result holds.

4 Simple Algorithms

4.1 A Simple Curve Offset Algorithm

In this section we present a simple offset algorithm for B-spline curves. The approach given here is essentially that of Cobb [2].

Let the curve to be offset be specified as a B-spline curve γ of order k:

$$\gamma(t) = \frac{\sum_{i=1}^{m} h_i \mathbf{E}_i B_{i,k}(t)}{\sum_{i=1}^{m} h_i B_{i,k}(t)}.$$

The offset curve of γ by the vector $\mathbf{V}$ relative to the frame $\mathcal{F}(t)$ is approximated by,

$$\hat{\mathbf{o}}(t) = \frac{\sum_{i=1}^{m} h_i \mathbf{E}'_i B_{i,k}(t)}{\sum_{i=1}^{m} h_i B_{i,k}(t)},$$

where $\mathbf{E}'_i$ is called the **Euclidian control point offset** of $\mathbf{E}_i$ with respect to curve γ, offset vector $\mathbf{V}$, and frame $\mathcal{F}$ and is given by:

$$\mathbf{E}'_i = \mathbf{E}_i + \mathcal{F}(t^*_i)\mathbf{V}, \tag{9}$$

with t^*_i the ith parametric **node** of the knot vector for $\gamma(t)$. We call (9) a **Euclidian control point offset operator.** (We discuss the selection of the frame $\mathcal{F}(t)$ in Section 6. This algorithm merely assumes that the frame can be evaluated at discrete points.)

We note that the offset curve approximation $\hat{\mathbf{o}}(t)$ has the same order and knot vector as $\gamma(t)$. The offset curve approximation also inherits the same set of homogeneous coordinates as the original curve.

The Euclidian projections of the control points of $\hat{\mathbf{o}}(t)$ are derived by offsetting the $\mathbf{E}_i$ of γ by the vector $\mathbf{V}$ relative to the local coordinate frame $\mathcal{F}(t)$ at a point on the curve γ that is associated with each $\mathbf{E}_i$. We use the parametric nodes of the knot vector for γ for this purpose (see Section 2)[1].

This simple algorithm gives exact results in the trivial case where the frame $\mathcal{F}(t)$ is constant. The offset curve is then simply a translation of the original curve.

In the general case it can be shown that a sequence of offset approximations can be made to converge to the exact offset curve with refinement of $\gamma(t)$ (see Appendix B). Therefore, although in general the offset of a rational B-spline curve can not be represented exactly by a rational B-spline, we

[1]The parametric nodes of a B-spline with **floating end conditions** are not all in the parametric domain of the B-spline. For such curves we can apply the above algorithm by first converting the curve to a B-spline with **open end conditions** by using B-spline refinement.

can approximate it to any degree of accuracy desired by suitably refining the curve being offset and then applying the above offset algorithm to the refined curve.

4.2 A Simple Sweep Algorithm

The following is a simple algorithm for approximating a sweep surface. The approach is again that of Cobb [2].

Let the axis curve of order k be specified as:

$$\mathbf{a}(u) = \frac{\sum_{j=1}^{m} h_j^a \mathbf{A}_j B_{j,k}^a(u)}{\sum_{j=1}^{m} h_j^a B_{j,k}^a(u)}.$$

Let the cross-section curve of order l be specified as:

$$\mathbf{c}(v) = \frac{\sum_{i=1}^{n} h_i^c \mathbf{C}_i B_{i,l}^c(v)}{\sum_{i=1}^{n} h_i^c B_{i,l}^c(v)}.$$

We approximate the surface generated by sweeping the cross-section $\mathbf{c}(v)$ along the axis curve $\mathbf{a}(u)$ using the local orientation frame $\mathcal{F}(u)$ by:

$$\hat{\boldsymbol{\sigma}}(u,v) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} h_j^a h_i^c \mathbf{A}'_{ij} B_{j,k}^a(u) B_{i,l}^c(v)}{\sum_{i=1}^{n}\sum_{j=1}^{m} h_j^a h_i^c B_{j,k}^a(u) B_{i,l}^c(v)},$$

where $\mathbf{A}'_{ij}$ is the Euclidian control point offset of $\mathbf{A}_j$ with respect to curve $\mathbf{a}(u)$, offset vector $\mathbf{C}_i$, and frame $\mathcal{F}$ and is given by:

$$\mathbf{A}'_{ij} = \mathbf{A}_j + \mathcal{F}(u_j^*)\mathbf{C}_i,$$

with u_j^* the jth parametric node of the knot vector for $\mathbf{a}(u)$.

We note that the sweep surface approximation inherits the axis curve's order and knot vector in the u direction and inherits the cross-section curve's order and knot vector in the v direction.

The surface structure created represents the tensor product B-spline, $\hat{\boldsymbol{\sigma}}(u,v)$, of equation (6) with:

$$\hat{\boldsymbol{\alpha}}_i(u) = \frac{\sum_j h_j^a \mathbf{A}'_{ij} B_{j,k}^a(u)}{\sum_j h_j^a B_{j,k}^a(u)}.$$

This sweep algorithm corresponds to computing the $\hat{\boldsymbol{\alpha}}_i(u)$ curves by the curve offset algorithm of Section 4.1. These curves can be made arbitrarily close to the exact offset curves, $\boldsymbol{\alpha}_i(u)$, with refinement of the axis curve. Thus, by (8), we can cause $\hat{\boldsymbol{\sigma}}(u,v)$ to be arbitrarily close to the true sweep surface by suitably refining the axis curve and then using the refined curve as input to the sweep algorithm.

5 Sweep Extensions

The general form of extension to sweeps as mentioned in Section 3 is:

$$\boldsymbol{\sigma}(u,v) = \mathbf{a}(u) + \boldsymbol{\mathcal{F}}(u)\mathbf{c}(u,v),$$

which allows the cross-section to deform as a function of u.

Isolines in v of such a surface then have the form:

$$\mathbf{o}(u) = \mathbf{a}(u) + \boldsymbol{\mathcal{F}}(u)\mathbf{V}(u).$$

These curves can be considered offset curves of $\mathbf{a}(u)$, where the offset vector is now deformable as a function of u.

We consider only deformations of the B-spline cross-section curve that can be expressed as deformations of its control polygon. Each projective point of the cross-section's control polygon is now considered a function of u with the form:

$$h_i^c(u)(\mathbf{C}_i(u),1). \tag{10}$$

The simplest method of extending the sweep algorithm of Section 4.2 is to evaluate the functions $h_i^c(u)$ and $\mathbf{C}_i(u)$ each time we offset a control point of the axis curve. The parametric point of evaluation is at the associated node of the axis curve's knot vector.

That is:

$$\hat{\boldsymbol{\sigma}}(u,v) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} h_j^a h_i^c(u_j^*)\mathbf{A}'_{ij}B_{j,k}^a(u)B_{i,l}^c(v)}{\sum_{i=1}^{n}\sum_{j=1}^{m} h_j^a h_i^c(u_j^*)B_{j,k}^a(u)B_{i,l}^c(v)},$$

where:

$$\mathbf{A}'_{ij} = \mathbf{A}_j + \boldsymbol{\mathcal{F}}(u_j^*)\mathbf{C}_i(u_j^*).$$

The

$$\hat{\boldsymbol{\alpha}}_i(u) = \frac{\sum_{j=1}^{m} h_j^a \mathbf{A}'_{ij} B_{j,k}^a(u)}{\sum_{j=1}^{m} h_j^a B_{j,k}^a(u)},$$

are now approximations to the curves given by:

$$\boldsymbol{\alpha}_i(u) = \mathcal{T}(u)(\mathbf{C}_i(u)).$$

These curves can be considered offset curves of $\mathbf{a}(u)$ by the (deformable) vector $\mathbf{C}_i(u)$ relative to the frame $\mathcal{F}(u)$.

Normally we would expect only to approximate the deformation of the cross-section in the sweep surface approximation by this approach. However a sequence of sweep surface approximations can be made to converge to the correct result with refinement of the axis curve as a preprocessing step (see Appendix B).

Two typical techniques for specifying this cross-section deformation are **profile curves** and **cross-section blending**. We discuss each approach in turn.

5.1 Profile Curves

We can allow the cross-section curve to undergo uniform scaling as it is swept along the axis curve. That is:

$$\boldsymbol{\sigma}(u, v) = \mathbf{a}(u) + \mathcal{F}(u)(g(u)\mathbf{c}(v)),$$

where the $h_i^c(u)$ and $\mathbf{C}_i(u)$ functions of equation (10) are given as:

$$h_i^c(u) = h_i^c \text{ and } \mathbf{C}_i(u) = g(u)\mathbf{C}_i.$$

Within the context of a geometric modeling system, we choose to express the scaling function $g(u)$ as a **profile curve**. A profile curve is a B-spline, specified in the xy plane, that represents the graph of an explicit function of x (i.e., the profile curve increases monotonically in x as a function of the curve parameter). See Figures 5 and 6.

Conceptually, we establish a mapping between a **normalized arc length measure** along the axis curve and the x coordinate along the profile curve.

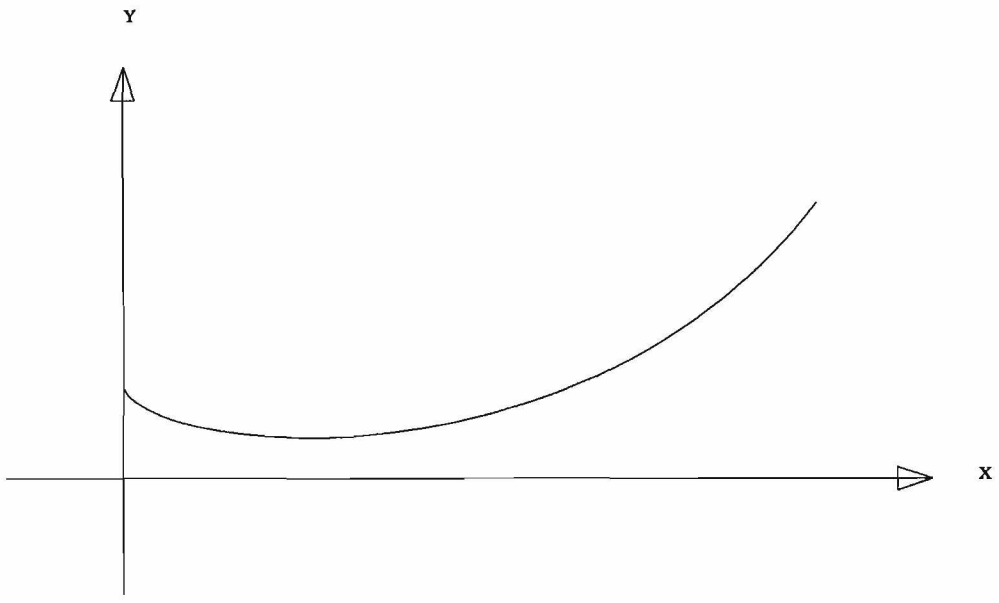

Figure 5: B-spline profile curve representing an explicit function of x.

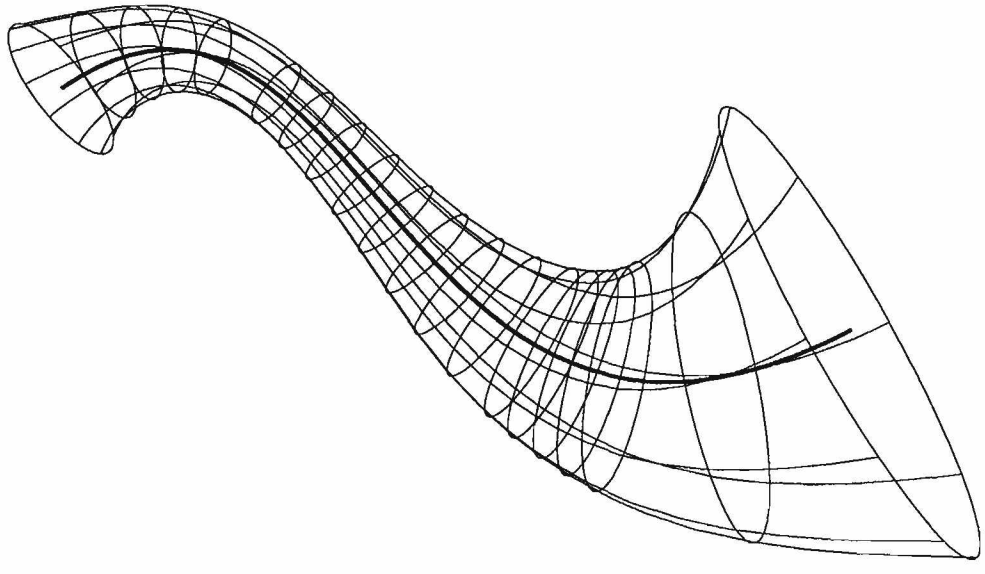

Figure 6: Sweep using the profile curve of Figure 5.

At each position as we sweep the cross-section along the axis curve, the y coordinate at the corresponding value of x on the profile curve, is used directly as the factor by which the cross-section is scaled.

In practice we establish the following mappings:

Map 1 Axis curve parameter value *to/from* normalized arc-length measure along the axis curve. The arc-length measure is normalized from 0 to 1 in the parametric domain of the axis curve. We denote this mapping as:

$$(\textbf{axis-param} \longleftrightarrow \textbf{normalized-axis-arc-length})$$

Map 2 Profile curve parameter value *to/from* normalized measure of profile curve x coordinate. The measure of the x coordinate is normalized from 0 to 1 in the parametric domain of the profile curve. We denote this mapping as:

$$(\textbf{profile-param} \longleftrightarrow \textbf{normalized-profile-x})$$

Map 3 Profile curve parameter value *to* actual y coordinate of the profile curve. We denote this mapping as:

$$(\textbf{profile-param} \longrightarrow \textbf{profile-y})$$

In an actual implementation, **Maps 1** and **2** can be approximated as invertible piecewise linear functions using table look up. The reason for the bi-directional nature of these two maps becomes apparent below. For **Map 3** we use the B-spline profile curve itself along with B-spline evaluation.

Normalized arc-length measures along the axis curve can be approximated using chord-length measures at evaluated sample points on the axis curve. In order that enough samples are taken to insure a reasonably accurate **Map 1**, the axis curve's knot vector can be refined using a curvature based refinement scheme as in [15]. Such a scheme effects the greatest refinement in parametric areas where the axis curve is geometrically most non–linear. Some fixed number of samples can then be taken on each parametric interval of the refined knot vector. The result is that most samples are concentrated in areas on the axis curve that are geometrically most non-linear.

This same knot vector refinement strategy can be applied in taking samples off the profile curve in order to obtain a reasonably accurate **Map 2**.

The $\mathbf{C}_i(u)$ of equation (10) are given by $g(u)\mathbf{C}_i$ where the scaling function $g(u)$ is computed for a given value of u on the axis curve by using the following composite map:

$$\begin{aligned}&(\textbf{axis-param} \longrightarrow \textbf{normalized-axis-arc-length}) \longrightarrow \\ &\quad (\textbf{normalized-profile-x} \longrightarrow \textbf{profile-param}) \longrightarrow \\ &\quad\quad (\textbf{profile-param} \longrightarrow \textbf{profile-y})\end{aligned}$$

See Figure 7.

5.1.1 Refinement of the Axis to the Geometry of the Profile

In general we need to refine the axis curve according to the geometry of the profile curve, in order that the resulting sweep surface approximation takes on the characteristics of the profile curve in the axis direction (see [10] and [7]). The resulting axis curve refinement is then used as input to the sweep-with-profile algorithm. Figures 8 through 10 illustrate an example where this refinement is very important. The tangent discontinuity in the profile curve is not properly reflected in the sweep surface approximation of Figure 9 because of the lack of refinement of the axis curve.

The refinement of the axis curve to the geometry of the profile curve can be accomplished as follows:

First we insure that the profile curve's knot vector is suitably refined using curvature based refinement (the refinement of the profile curve's knot vector for purposes of attaining a reasonable **Map 2** could be used). We then map the knots of the refined profile curve into the knot vector of the axis curve by using the following composite map:

$$\begin{aligned}&(\textbf{profile-param} \longrightarrow \textbf{normalized-profile-x}) \longrightarrow \\ &\quad (\textbf{normalized-axis-arc-length} \longrightarrow \textbf{axis-param})\end{aligned}$$

The mapped knots are then merged into the axis curve's knot vector and the axis curve refined.

In merging the mapped knots from the profile curve, we must be careful not to allow the newly added knots to bunch up with existing knots in the axis curve's knot vector. Otherwise, we could introduce potential value or tangent discontinuities not present in either the axis or profile curves. If a

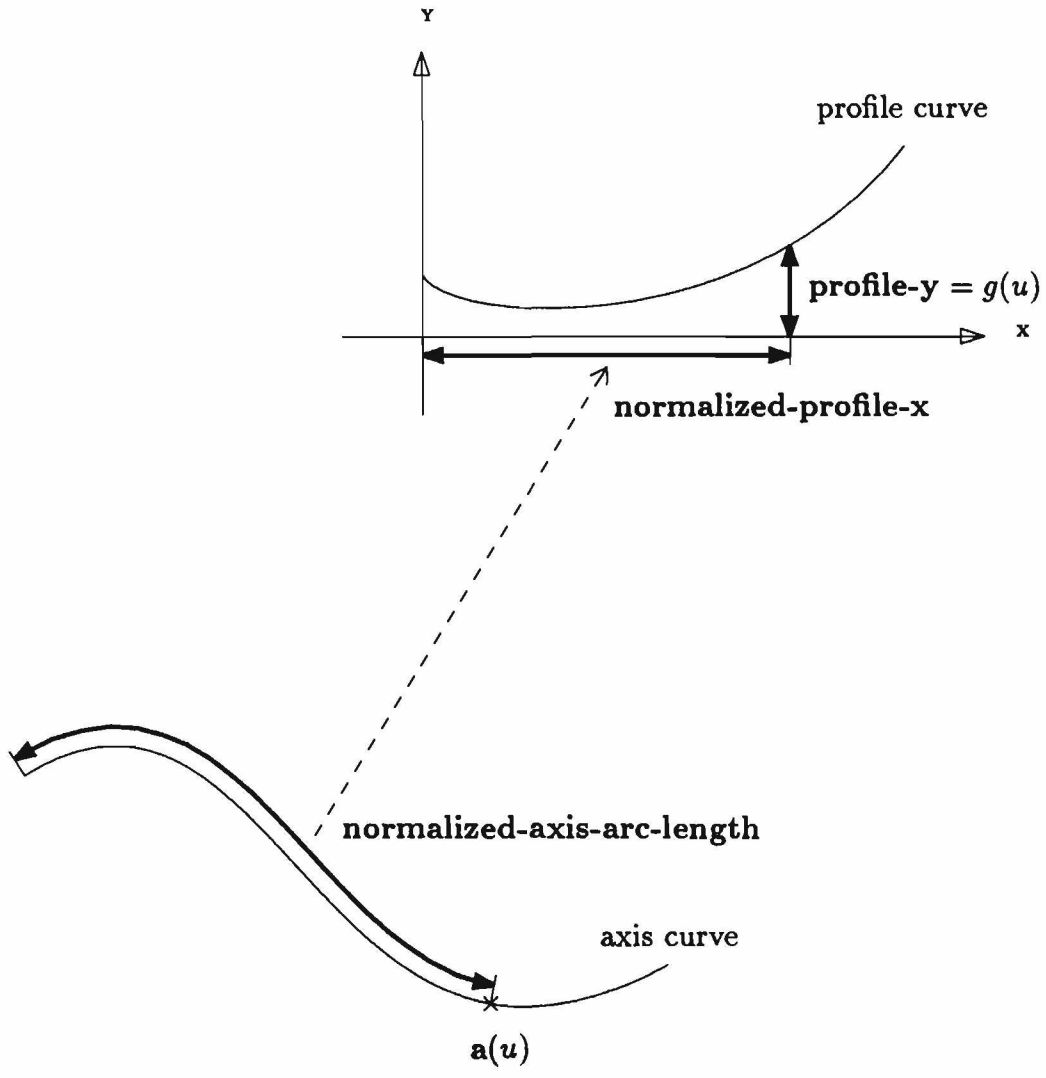

Figure 7: Cross-section scale factor derived from the profile curve for a given point $\mathbf{a}(u)$ on the axis curve.

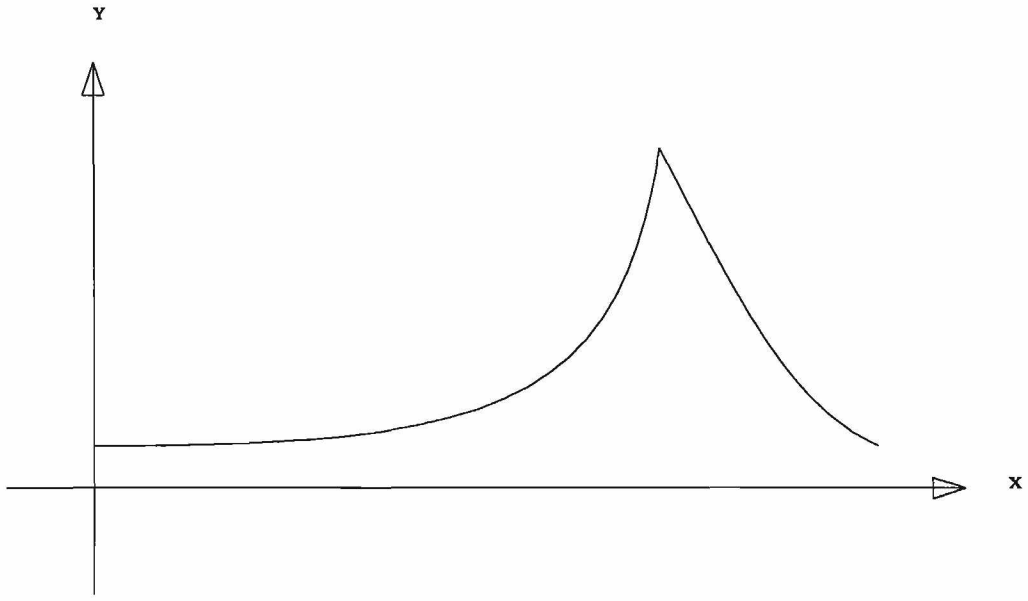

Figure 8: Profile curve with a tangent discontinuity.

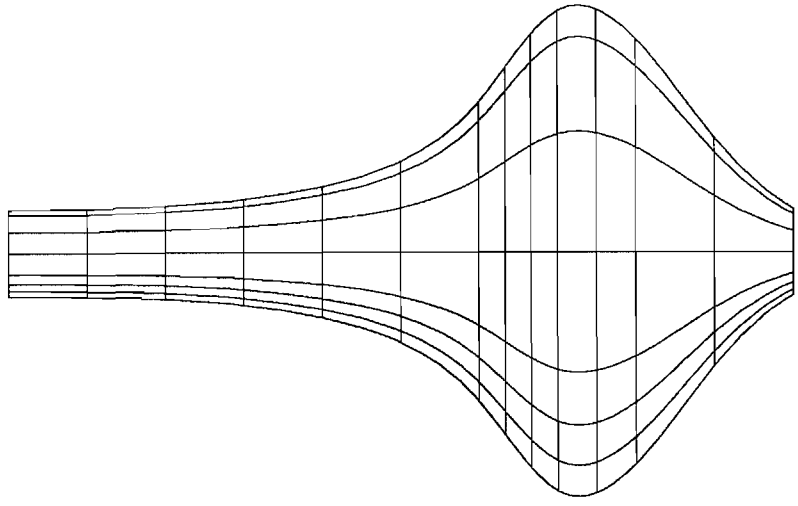

Figure 9: Orthographic view of a surface generated by sweeping a cross-section along a linear axis curve. Variable scaling of the cross-section is given by the profile curve of Figure 8. The tangent discontinuity in the profile curve is not properly reflected in the sweep surface because of the lack of refinement of the axis curve.

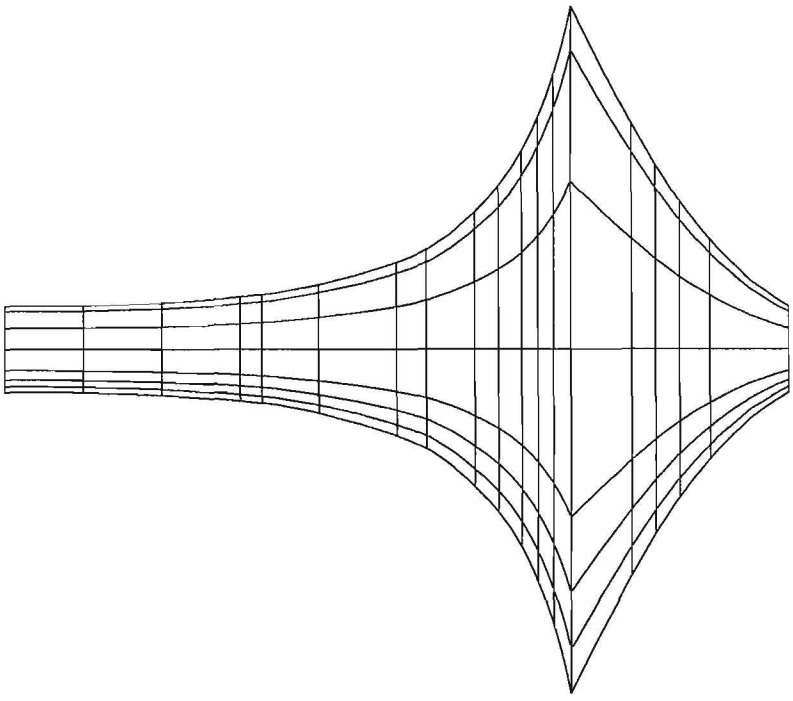

Figure 10: Appropriate refinement of the axis curve yields a much better result.

mapped knot is within a given radius, ϵ_1, to one already in the axis curve's knot vector, it is not added.

Keeping in mind that we only want to insure that the axis curve undergoes refinement in areas that correspond to interesting areas of the profile curve (where the scale values change very non-linearly), it is generally sufficient to add only a single (non-multiple) knot for each (possibly multiple) knot mapped from the profile curve's knot vector.

This is not the case for knots giving rise to potential tangent discontinuities in the profile curve however. In this case we wish to insure that the potential tangent discontinuity is reflected in the refined axis curve (and hence in the resulting sweep approximation).

Let $\mathbf{order}^p$ be the order of the profile curve and $\mathbf{order}^a$ be the order of the axis curve. A knot of multiplicity $\mathbf{order}^p - 1$ in the profile curve's knot vector should give rise to the addition of a knot of multiplicity $\mathbf{order}^a - 1$ in the axis curve's knot vector. To prevent bunching up of knots (resulting in possible spurious value discontinuities), if the knot to be added is within a given radius, ϵ_2, of an existing knot in the axis curve's knot vector, then the existing knot should have its multiplicity raised to $\mathbf{order}^a - 1$ if necessary (rather than adding a knot at a new parametric value).

In general we want ϵ_2 to be smaller than ϵ_1. Any noticeable effects of tangent discontinuities in the profile curve on cross-section curve scaling (such as local maxima or minima of the scaling function) need to occur, in the sweep process, very close to any potential tangent discontinuities introduced into the axis curve. The placement of mapped profile knots with multiplicity less than $\mathbf{order}^p - 1$ is less critical.

5.2 Cross-Section Blending

Cross-section blending conceptually involves sweeping a deformable cross-section curve along the axis, where at any given instant the cross-section is a blend of a specified set of B-spline cross-section curves. See Figures 11 and 12.

Without loss of generality, we assume that all the B-spline cross-section curves to be blended are defined over a common domain and common set of B-spline basis functions[2], i.e.,

[2]By using B-spline degree raising [4], positive affine transformations of knot vectors,

$$\mathbf{c}_k(v) = \frac{\sum_i w_{k,i}\mathbf{Q}_{k,i}B_i(v)}{\sum_i w_{k,i}B_i(v)}.$$

We only consider cross-section curve blending of a form expressible as the blending of corresponding control points of the cross-sections, with blending of control points for rational curves taking place in projective space[3]. That is, the $h_i^c(u)$ and $\mathbf{C}_i(u)$ functions of equation (10) are given as:

$$h_i^c(u) = \sum_k \lambda_k(u) w_{k,i} \text{ and } \mathbf{C}_i(u) = \sum_k \lambda_k(u)\mathbf{Q}_{k,i} \ ,$$

where the $\lambda_k(u)$ are the blending functions.

One approach is to use a simple linear interpolation scheme. Along with each cross-section specified in the blending set $\{\mathbf{c}_k(v)\}_k$ a value s_k corresponding to a normalized arc-length measure along the axis curve is given. This value is used to indicate at what point in the sweep process the sweep surface cross-section should most closely approximate the given cross-section. (We assume the s_k are given in ascending order.)

For a given value of u let $s(u)$ be the normalized arc-length measure along the axis curve at u (using **Map 1** of the previous section) and let s_k, s_{k+1} be such that $s_k \leq s(u) \leq s_{k+1}$.

Then we compute $h_i^c(u)$ and $\mathbf{C}_i(u)$ of equation (10) as:

$$\begin{aligned} h_i^c(u) &= \lambda_k(u)w_{k,i} + \lambda_{k+1}(u)w_{k+1,i} \quad \text{and} \\ \mathbf{C}_i(u) &= \lambda_k(u)\mathbf{Q}_{k,i} + \lambda_{k+1}(u)\mathbf{Q}_{k+1,i} \ , \end{aligned}$$

where $\lambda_k(u) = \frac{s_{k+1}-s(u)}{s_{k+1}-s_k}$ and $\lambda_{k+1}(u) = \frac{s(u)-s_k}{s_{k+1}-s_k}$.

---

and B-spline refinement [3], we can reformulate any set of B-spline curves to be defined over a common order and knot vector.

[3]Blending in Euclidian space of rational B-spline curves, having differing sets of homogeneous coordinates, in general, involves an appreciable increase in the complexity of the result (expressed as a rational B-spline). This is because the rational terms in the blend would need to be expressed over a common denominator. For this reason we blend such curves in projective space by blending their corresponding control points.

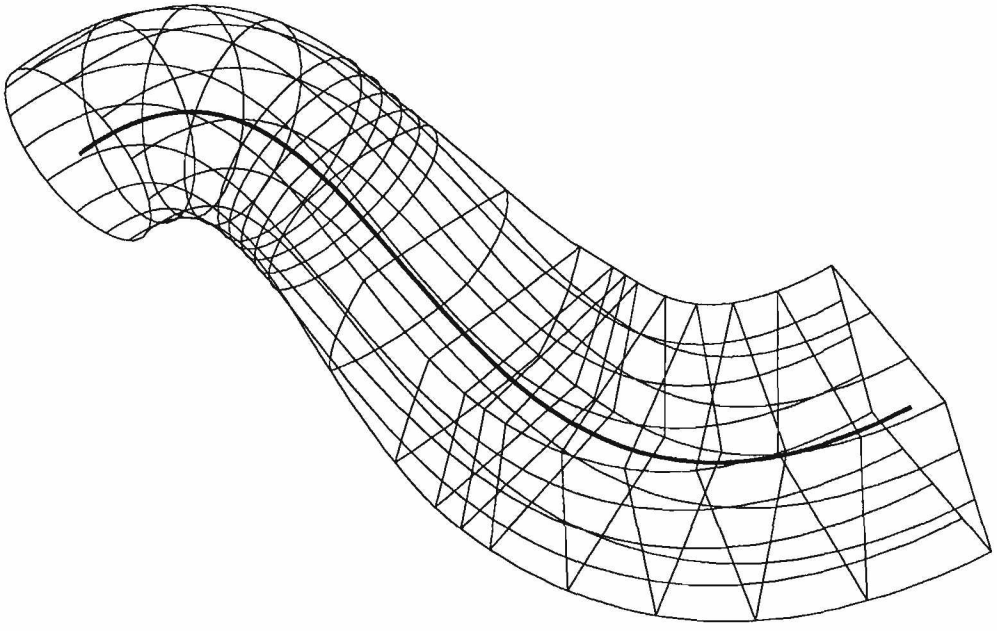

Figure 11: Sweep using cross-section blending.

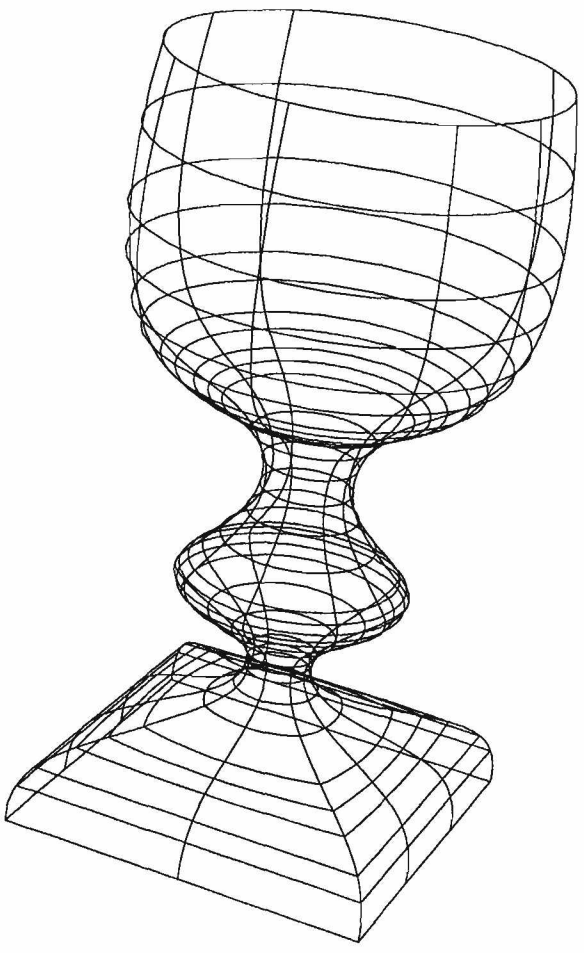

Figure 12: Sweep along a linear axis using both a profile curve and cross-section blending. Model by Sabine Coquillart.

5.2.1 Refinement of the Axis to Cross-Section Placement

In general we must refine the axis curve according to the placement of cross-sections, given by the s_k, in order that the cross-sections of the resulting sweep surface approximation take on the characteristics of the specified cross-sections. We map each of the s_k into the parameter space of the axis curve by using the (**normalized-axis-arc-length** $\longrightarrow$ **axis-param**) map of Section 5.1. We can then add some number of equally spaced knots in the interval around the parametric point in the axis curve's knot vector. We must again be careful not to allow knots being added to bunch up with knots already in the axis curve's knot vector (see Section 5.1.1). The refined axis curve is then used as input to the sweep algorithm.

5.2.2 Need for More General Blending

Our approach has thus far been to give sweep approximations that can be made arbitrarily close to the true sweep surface with refinement of the axis curve. The piecewise linear nature of the blending scheme used above however, becomes apparent with this refinement. The resulting sweep surface approximation begins to exhibit piecewise linear behavior in the u direction. Practical use of this simple blending scheme relies on the smoothing properties of the tensor product B-spline approximation and avoidance of too much refinement of the axis curve. More general blending schemes are required to help alleviate this problem.

6 The Orientation Problem

We now address the problem of specifying the local coordinate frame, $\mathcal{F} = \left[\begin{array}{ccc} \mathbf{X} & \mathbf{Y} & \mathbf{Z} \end{array} \right]$, along the curve to be offset (or the axis curve in the case of a sweep). This problem was referred to as the **orientation problem** by Shani and Ballard [16].

Figures 13 through 16 illustrate that we may be interested in many potential solutions to this problem. Figure 13 shows a sweep of a gear cross-section along a linear axis to produce a straight gear surface. Here the frame $\mathcal{F}$ is constant. We classify the result as a **translational sweep** or a **linear extrusion**. Figure 14 shows the same cross-section swept along the same axis, but this time the frame $\mathcal{F}$ is chosen to produce a helical gear.

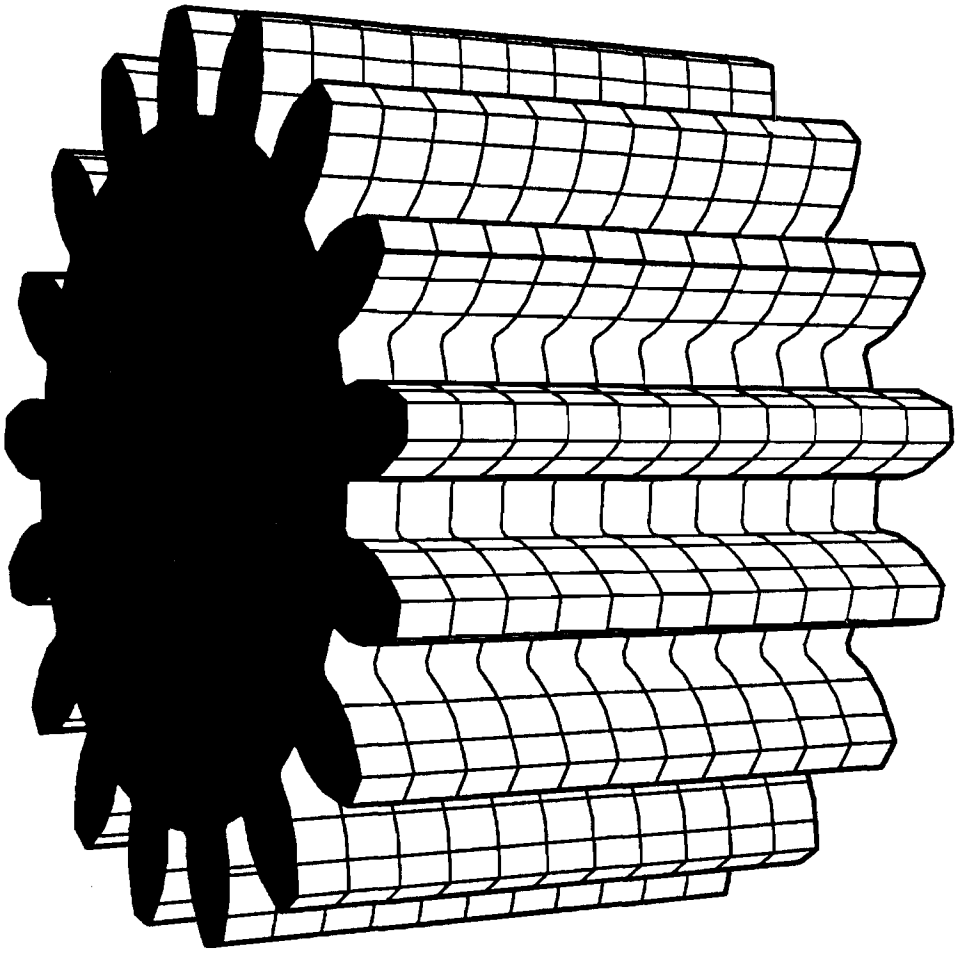

Figure 13: Straight gear surface produced as a sweep along a linear axis. The local coordinate frame is constant along the axis curve.

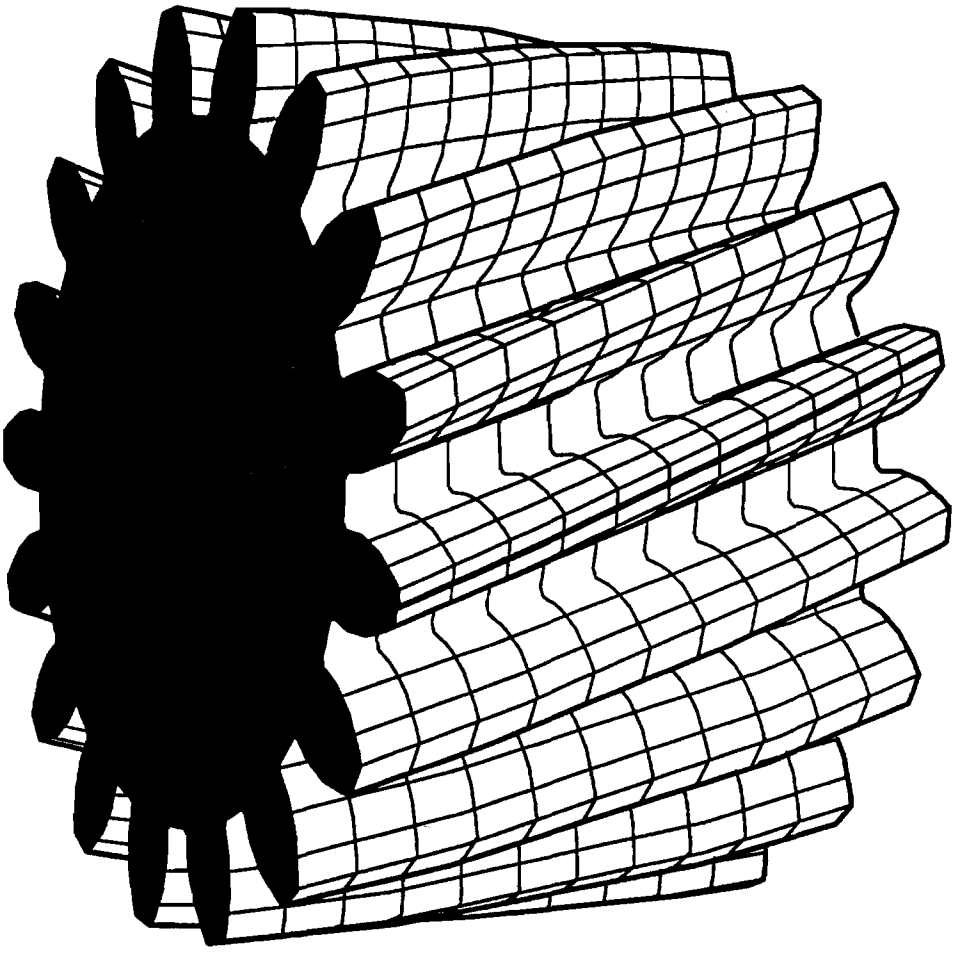

Figure 14: Helical gear surface produced as a sweep along a linear axis. The local coordinate frame rotates along the axis curve.

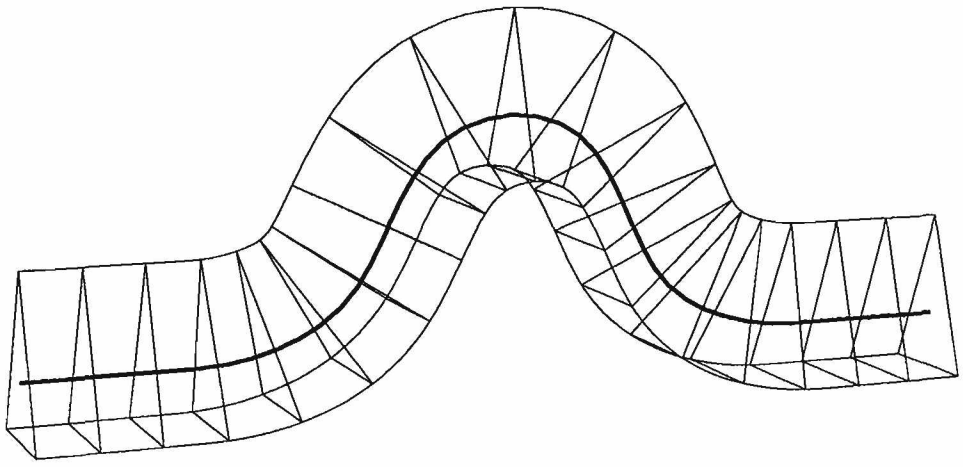

Figure 15: Sweep of a triangular cross-section along a planar axis curve. The local coordinate frame used is related to the Frenet frame along the axis curve.

Figure 15 shows a triangular cross-section swept along a planar axis curve to produce a fairly intuitive result. Here the frame $\mathcal{F}$ is related to the Frenet frame along the axis curve, but is defined so as to be unaffected by vanishing curvature. The same axis and cross-section curves are used in Figure 16 to produce a different shape. Here the frame $\mathcal{F}$ is constant, that is the cross-section is kept at a fixed orientation; this is another example of a translational sweep.

In this paper we do not deal with the issue of how to specify the orientation frame in general. Rather the orientation information is either derived from the axis curve (or curve being offset), or implied by a category of sweep (e.g., helical sweep, translational sweep).

One justification for this approach is that it simplifies the interface to the sweep and offset operators. Equally important, there are many cases where there is a strong intuitive notion of what the behavior of an offset to a particular curve (or a sweep using a particular curve as an axis) should be. In these cases we want to free the user from detailed specification of orientation information. Even where this is not the case (for very general 3-space curves), establishing a default orientation along the curve may be

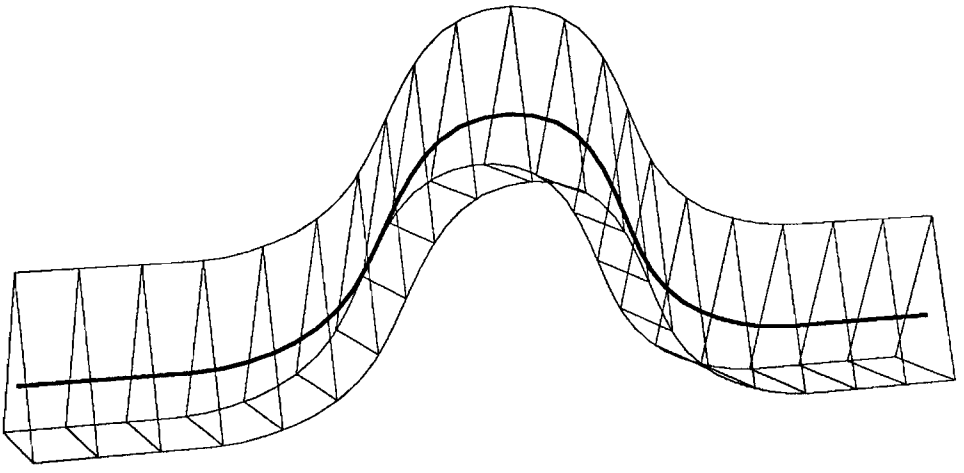

Figure 16: Translational sweep using the same axis and cross-section curves used in Figure 15. The local coordinate frame is constant for the entire sweep.

desirable. Such a default might be used, for example, as a reference for establishing more general orientation specifications along the curve.

What are likely candidates for a default orientation along a curve?

6.1 Planar Curves

In the case of planar curves with non-vanishing curvature, the Frenet frame forms a fairly intuitive default orientation along the curve, i.e., we take $\mathcal{F} = \begin{bmatrix} \mathbf{N} & \mathbf{B} & \mathbf{T} \end{bmatrix}$.

One problem with the Frenet frame however is that the principal normal vanishes at inflection points and can reverse its direction there. The result is shown in Figures 17 and 18 for offsets and sweeps with respect to the Frenet frame.

One way of formulating a solution to this problem for planar curves is as follows (see [11]): Let $\mathcal{N}_p$ be the normal to the plane of the curve and let $\mathbf{Z} = \mathbf{T}$, $\mathbf{Y} = \mathcal{N}_p$, and $\mathbf{X} = (\mathbf{Y} \times \mathbf{Z}) = (\mathcal{N}_p \times \mathbf{T})$. This frame differs from the Frenet frame for planar curves in that the vector $\mathbf{X}$ does not reverse its direction at inflection points. The vector $\mathbf{X}$ is also well defined on embedded

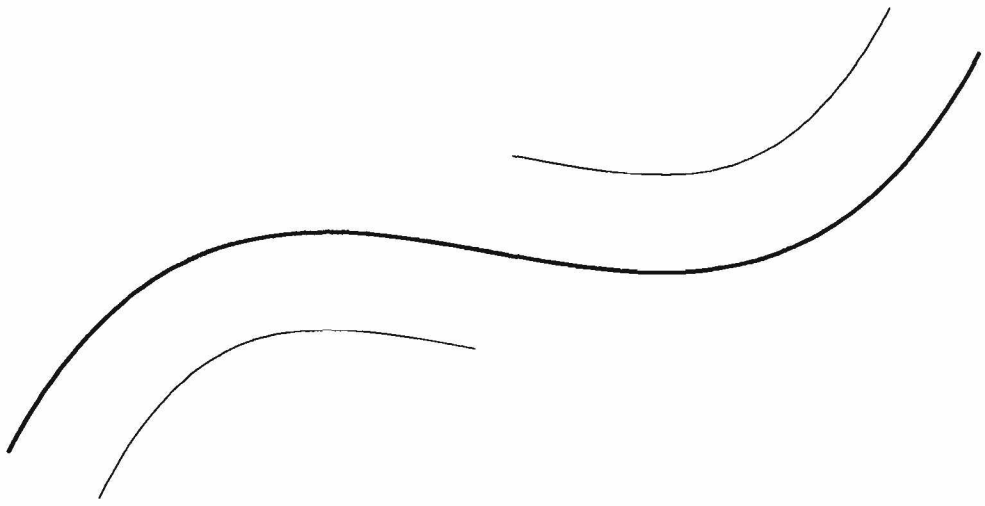

Figure 17: Offset relative to the Frenet frame of a curve with an inflection point.

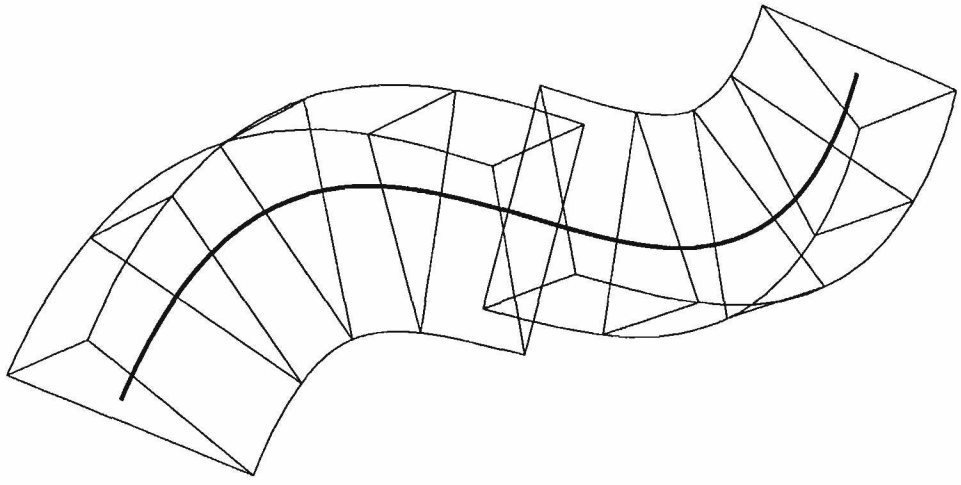

Figure 18: Sweep relative to the Frenet frame along an axis curve with an inflection point.

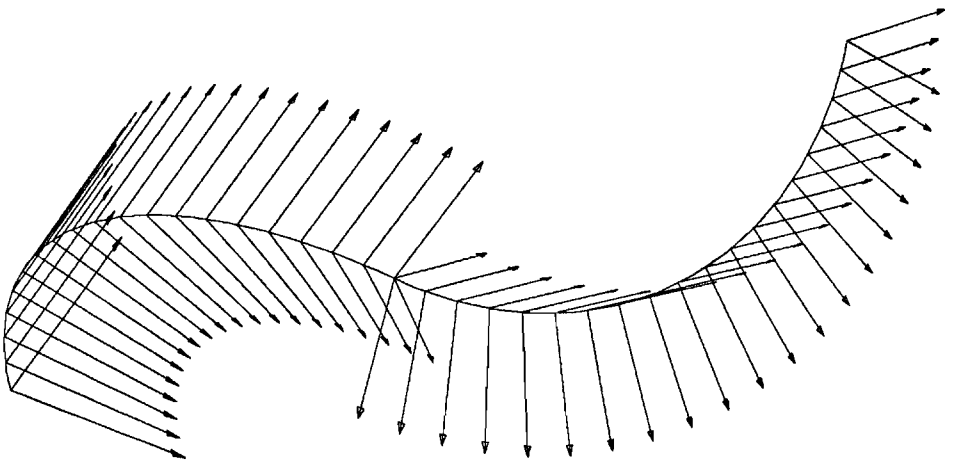

Figure 19: The curve above is formed from two arcs joined so as to be unit tangent continuous. The planes in which the arcs lie are inclined by 45 degrees relative to one another. The plane normals are indicated by the upper set of arrows along the curve.

linear segments, whereas the principal normal is not.

We note that this frame simply rotates about the plane normal $\mathcal{N}_p$. Another way of formulating this solution is then:

Let $\mathbf{X}(u_0) = \mathcal{N}_p \times \mathbf{T}(u_0)$, $\mathbf{Y}(u_0) = \mathcal{N}_p$, and $\mathbf{Z}(u_0) = \mathbf{T}(u_0)$, give the frame at the first point, u_0, of the curve. We compute the frame at a general point u on the curve by rotating the frame at u_0 about the plane normal $\mathcal{N}_p$, by the amount necessary to rotate $\mathbf{T}(u_0)$ into $\mathbf{T}(u)$.

We can generalize this notion slightly by allowing the frame at u_0 to be any orthonormal frame $\mathcal{F}(u_0)$ such that $\mathbf{Z}(u_0) = \mathbf{T}(u_0)$.

6.2 Piecewise Planar Curves

The solution presented above is readily generalized to the case of piecewise planar curves. The problem here is that the plane normal of the curve may change discontinuously giving rise to a discontinuous change in the local coordinate frame (see Figure 19).

This problem can be corrected by adopting the approach presented above

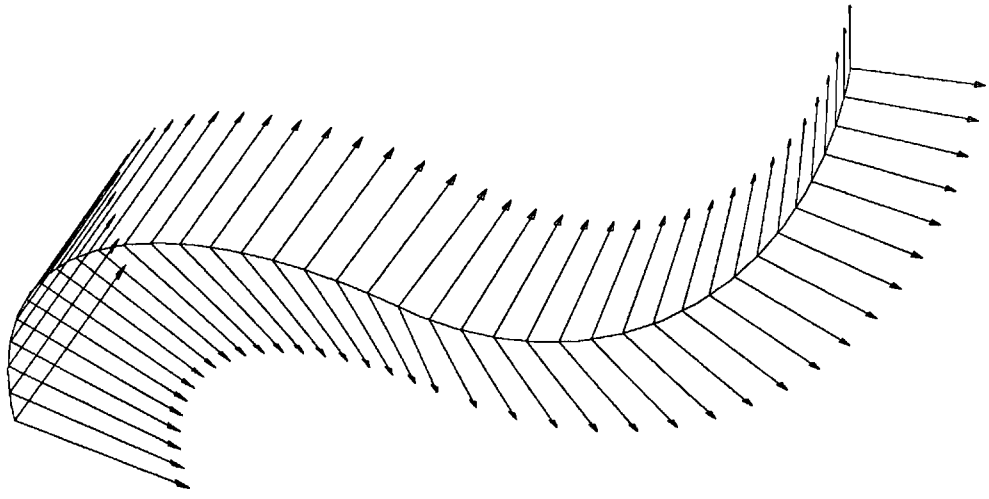

Figure 20: The rule given in Section 6.2 results in a well behaved local coordinate frame along the same curve used in Figure 19.

in a piecewise manner. Let $\mathcal{N}_i$ and $\mathcal{N}_{i+1}$ be the plane normals for the curve over the planer intervals $[u_i, u_{i+1})$ and $[u_{i+1}, u_{i+2})$, respectively. Then for u in the *closed* interval $[u_i, u_{i+1}]$, set $\mathcal{F}(u) = \mathcal{R}_i(u)(\mathcal{F}(u_i))$, where $\mathcal{R}_i(u)$ is a rotation about $\mathcal{N}_i$ by the amount necessary to rotate $\mathbf{T}(u_i)$ into $\mathbf{T}(u)$. For u in $[u_{i+1}, u_{i+2}]$, set $\mathcal{F}(u) = \mathcal{R}_{i+1}(u)(\mathcal{F}(u_{i+1}))$, where $\mathcal{R}_{i+1}(u)$ is a rotation about $\mathcal{N}_{i+1}$ by the amount necessary to rotate $\mathbf{T}(u_{i+1})$ into $\mathbf{T}(u)$. By this rule the frame at the beginning of one planar interval is identical to the frame at the end of the previous interval. Thus the frame changes continuously between planar pieces. Figure 20 shows the result of applying this rule.

Given that we have an intuitive default orientation for planar and piecewise planar curves, we ask how to extend this notion to general 3-D curves.

6.3 The Quasi-Normal Frame

One approach is to use an adjusted Frenet frame. Adjustments can be made at discrete points on the curve where the Frenet frame is poorly behaved: points of vanishing curvature (including the start and end of embedded linear

segments) and points at which the normal is discontinuous. Coquillart [6] uses such an approach by defining an orientation frame for unit tangent continuous space curves with reference to a vector called the **quasi-normal**.

The quasi-normal is a unit length vector in the curve's normal plane that can be thought of as the curve's principal normal corrected for inflection points, discontinuities of the normal, and embedded linear segments. The corrections for inflection points and discontinuities are made by keeping track of an angle θ between the principal normal and the quasi-normal as measured in the normal plane. The angle is initialized to zero at the beginning of the curve. At inflection points, π is added to θ. At discontinuities of the normal the angle between the left and right hand limits of the principal normal is subtracted from θ. The quasi-normal on an embedded linear segment is defined so as to be constant.

The quasi-normal frame is then $\mathcal{F} = \begin{bmatrix} \mathbf{Q}^N & \mathbf{Q}^B & \mathbf{T} \end{bmatrix}$ where $\mathbf{Q}^N(u)$ is the quasi-normal, $\mathbf{T}(u)$ is the unit tangent, and $\mathbf{Q}^B(u) = \mathbf{T}(u) \times \mathbf{Q}^N(u)$.

For planar and piecewise planar curves the quasi-normal frame is identical to the results obtained by the rotation technique discussed above (Sections 6.1 and 6.2).

On any portion of a general 3-space curve without discontinuities of the normal or vanishing curvature, the quasi-normal frame rotates with the Frenet frame. For non-planar curves with areas of high torsion, sweeps or offsets with respect to such a frame can yield very unintuitive results.

Consider an offset of the curve shown in Figure 21. This curve is represented as a planar cubic Bézier curve (i.e., a single polynomial piece). The curve has an inflection point that the quasi-normal computation can correct for (see Figure 22). (We assume that finding the location of the inflection point poses no problem; in general, however, this *is* a problem.)

It is easy to prove that a cubic polynomial can not have an inflection point unless it is a planar curve (see Appendix C). So if we move one of the control points of this curve out of the plane, the inflection point must disappear. The resulting curve may be only slightly non-planar (if we don't move the control point very far out of the plane). In this case, the reasonable expectation is that the resulting offset curve should be only slightly non-planar as well. If we define the offset with respect to a frame that rotates with the Frenet frame on this curve (such as the quasi-normal frame) this is not the case. The result is shown in Figure 23.

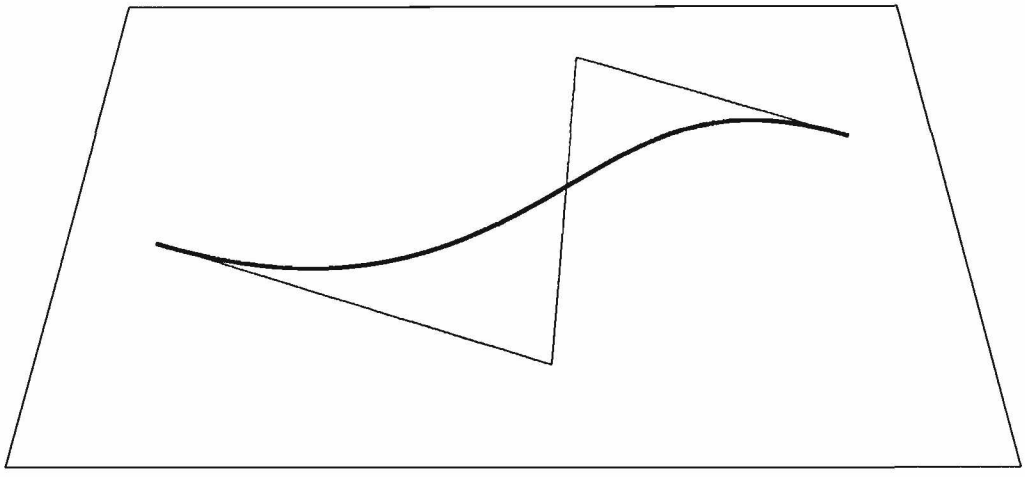

Figure 21: A planar cubic Bézier curve with an inflection point. (Perspective view.)

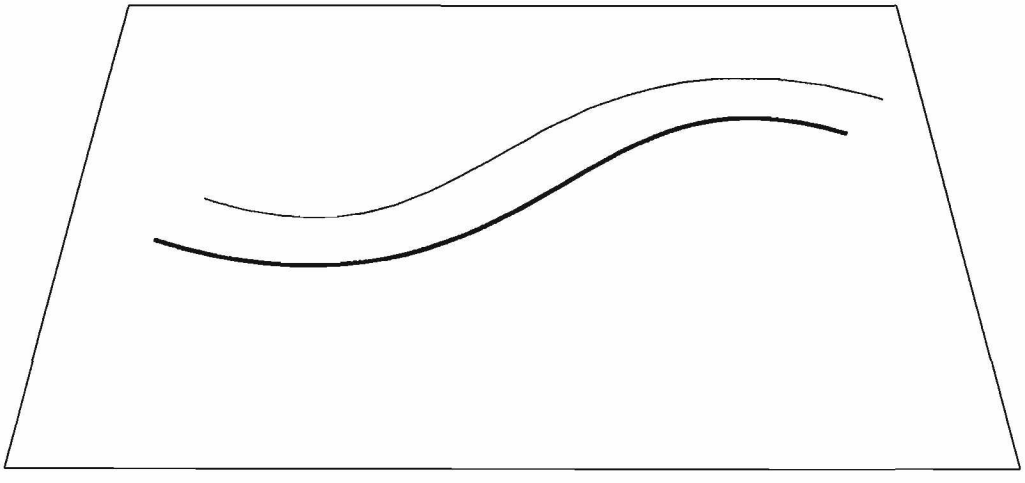

Figure 22: An offset of the curve of Figure 21 relative to the quasi-normal frame.

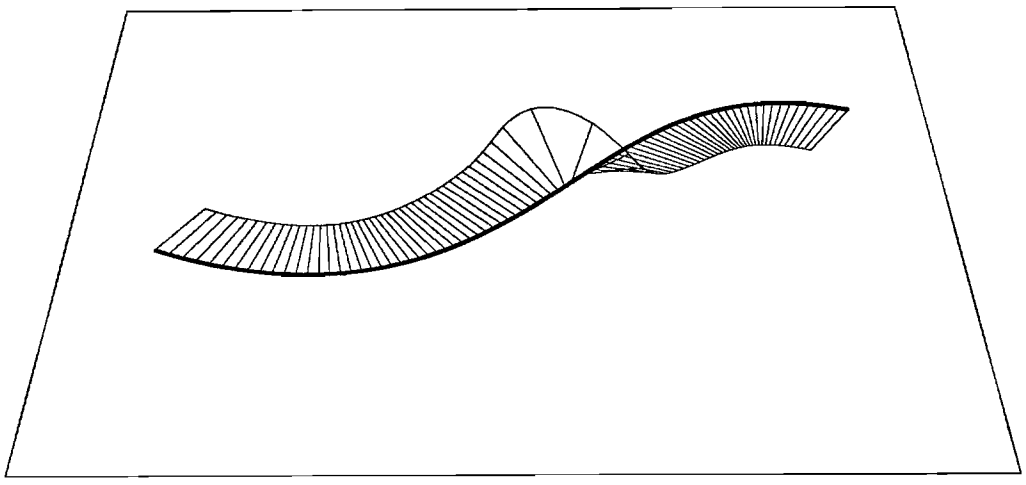

Figure 23: Offset relative to the quasi-normal frame of a slightly non-planar version of the curve of Figure 21.

The problem is that when we removed the inflection point, by making the curve non-planar, we introduced an area of very high torsion. Torsion can be considered a measure of the tendency of the curve's principal normal to rotate about the curve's tangent. The quasi-normal formulation can make corrections to the curve's normal at discrete points but can not correct for this tendency of the the Frenet frame to rotate around the curve's tangent in such areas.

6.4 Rotation Minimizing Frames

In the last section we saw that orientation frames defined with respect to the Frenet frame (or that rotate with the Frenet frame) can give rise to unwanted rotations about the tangent vector for general 3-space curves. In this section, we discuss orientation frames, called **rotation minimizing frames**, which exhibit *no* rotation about the tangent.

Rotation minimizing frames were introduced using an algorithmic definition by Shani and Ballard [16]. Klok [14] defined rotation minimizing frames in terms of the solution of a differential equation with initial conditions.

We state a definition along the lines of the one given by Shani and Ballard

by reference to a **rotation minimizing normal.**

6.4.1 Definition of Rotation Minimizing Frames

Let:

- $\boldsymbol{\gamma}(u)$ be a parametrically defined curve,
- $[u_{\min}, u_{\max}]$ be the parametric domain of $\boldsymbol{\gamma}(u)$,
- $\mathbf{T}(u)$ be the unit tangent function for $\boldsymbol{\gamma}(u)$,
- $\mathbf{q}$ be any unit length vector in the plane perpendicular to $\mathbf{T}(u_{\min})$,
- u' be a fixed but arbitrary value of u in the domain of $\boldsymbol{\gamma}(u)$,
- $\{p_i\}$ be a sequence of partitions of the interval $[u_{\min}, u']$ with the form:

$$u_{\min} = u_{i,1} < u_{i,2} < \cdots < u_{i,n_i} = u'$$

- $\|p_i\| = \max(u_{i,2} - u_{i,1},\ u_{i,3} - u_{i,2},\ \ldots)$ be the **max norm** of p_i.

We define $\mathbf{Q}_i(u')$ in accordance with:

$$\mathbf{Q}_i(u_{\min}) = \mathbf{q} \quad \text{and} \quad \mathbf{Q}_i(u_{i,j+1}) = \mathcal{R}_{u_{i,j}}(\mathbf{Q}_i(u_{i,j}))$$

where $\mathcal{R}_{u_{i,j}}$ is the transformation that rotates $\mathbf{T}(u_{i,j})$ into $\mathbf{T}(u_{i,j+1})$ in the plane spanned by $\mathbf{T}(u_{i,j})$ and $\mathbf{T}(u_{i,j+1})$. If $\mathbf{T}(u_{i,j})$ and $\mathbf{T}(u_{i,j+1})$ are linearly dependent, then $\mathcal{R}_{u_{i,j}}$ is the identity transformation.

The **rotation minimizing normal** of a curve $\boldsymbol{\gamma}(u)$ seeded by the vector $\mathbf{q}$, is denoted by $\mathbf{M}_q^N(u)$, and is defined as follows: if there exists a $\mathbf{Q}(u')$ such that for any arbitrary sequence $\{p_i\}$, as above, we have:

$$\lim_{i\to\infty} \|p_i\| = 0 \ \Rightarrow\ \lim_{i\to\infty} \mathbf{Q}_i(u') = \mathbf{Q}(u')$$

then $\mathbf{M}_q^N(u') \equiv \mathbf{Q}(u')$.

The **rotation minimizing binormal** of a curve $\gamma(u)$ seeded by the vector $\mathbf{q}$ is defined as:

$$\mathbf{M}_q^B(u) = \mathbf{T}(u) \times \mathbf{M}_q^N(u).$$

$\left[\mathbf{M}_q^N(u) \quad \mathbf{M}_q^B(u) \quad \mathbf{T}(u) \right]$ is referred to as a **rotation minimizing frame.**

6.4.2 Properties of Rotation Minimizing Frames

A rotation minimizing frame on a piecewise planar curve is equivalent to an orientation frame resulting from applying the technique of Section 6.2. In fact the definition of a rotation minimizing frame, as stated above, can be seen as a direct extension of the rotation technique for piecewise planar curves of Section 6.2. We apply the technique by considering a general 3-space curve to be composed of a series of infinitesimal planar pieces.

The rotation minimizing normal for a value of $u = u'$ undergoes instantaneous rotation about the curve's binormal at u'. This implies that the component of change of $\mathbf{M}_q^N(u')$ that lies in the plane perpendicular to $\mathbf{T}(u')$ must be zero (since $\mathbf{M}_q^N(u')$ and the axis about which it is instantaneously rotating both lie in this same plane). The component of rotation of $\mathbf{M}_q^N(u')$ about the tangent is therefore zero.

The above definition can be thought of in terms of a series of steps that rotate the frame as a whole through the direct angle between one tangent, $\mathbf{T}(u_{i,j})$, and the next, $\mathbf{T}(u_{i,j+1})$, at each step. This is the minimum possible rotation necessary to realign the $\mathbf{Z}$ axis of the frame with the curve's tangent at the new position $u_{i,j+1}$. As the max norm of the partitions approaches zero, the frame undergoes the minimal instantaneous rotation possible such that one axis is always aligned with the curve's tangent (see Figure 24). See Klok [14] for a more formal discussion.

Finally we note that on any *planar* portion of a curve without vanishing curvature, a rotation minimizing frame rotates with the Frenet frame. In other words, any vector which is fixed relative to a rotation minimizing frame is fixed relative to the Frenet frame on such portions of a curve.

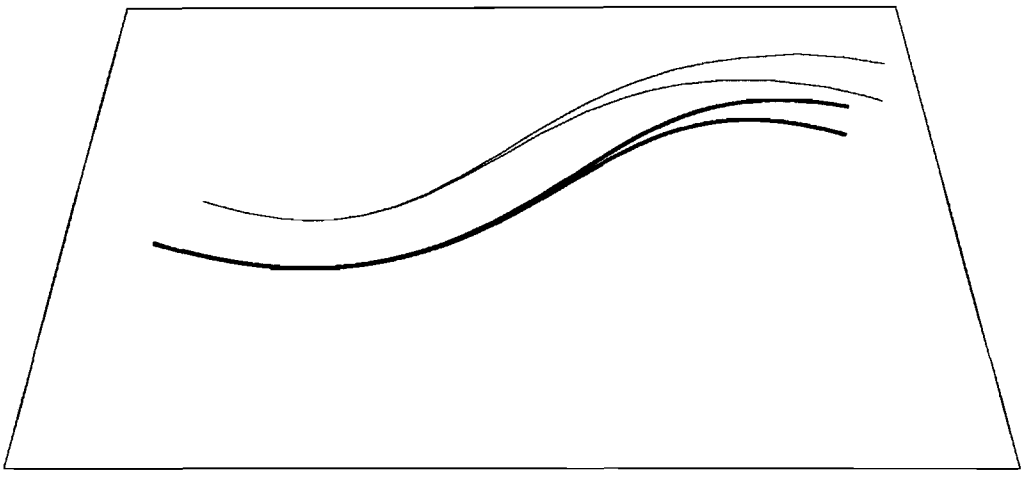

Figure 24: Offsets of the curves of Figures 21 and 23 relative to a rotation minimizing frame. The lower boldface curve is a planar cubic Bézier. The upper boldface curve is a slightly non-planar version of the same curve. The offset of the planar curve is the same as obtained by using the quasi-normal frame. The offset of the non-planar curve is much more intuitive than the offset relative to the quasi-normal frame shown in Figure 23.

6.4.3 Approximations of Rotation Minimizing Frames

In Section 7.2 we are interested in evaluating rotation minimizing frame functions at discrete points on a B-spline curve. In general, these function values can only be approximated. The approximations are made in a manner suggested by the definition for rotation minimizing frames given above.

We can choose any unit length vector in the normal plane at the first point of the curve, to serve as the initial value for the rotation minimizing normal. We then take small steps along the curve, sampling the tangent function and rotating the previous rotation minimizing normal approximation at each step. The points at which we sample must include the points where we desire the approximations for the rotation minimizing normal.

The number of additional steps required for a good approximation, on a given parametric interval, depends on the planarity of that interval. For piecewise planar curves we need only add additional sample steps at the bounds of the parametric intervals. The results for such curves are exact as this computation corresponds to the technique of Section 6.2.

For intervals that are non-planar, a number of samples interior to the interval are computed based on a measure of the planarity of that part of the curve's control polygon with control points blended on that interval.

7 Sweep Algorithms

In this section, we discuss: **translational sweeps**, **rotation minimizing frame sweeps** and **profile product sweeps**. Translational sweeps, discussed first, are generated by simple application of the sweep algorithm of Section 4.2. In Section 7.2 we present specialized offset and sweep algorithms using the rotation minimizing frames of Section 6.4. In Section 7.3 we discuss a sweep paradigm that generalizes surface of revolution.

7.1 Translational Sweeps and Linear Extrusions

Sweeps as defined by equation (1) with constant orientation frames $\mathcal{F}$ can be classified as **translational sweeps** (e.g., Figure 16). The $\boldsymbol{\alpha}_i(u)$ of equation (4) become simple translations of the axis curve and are computed exactly by the simple sweep algorithm of Section 4.2. This algorithm therefore computes these sweeps exactly (by (7)).

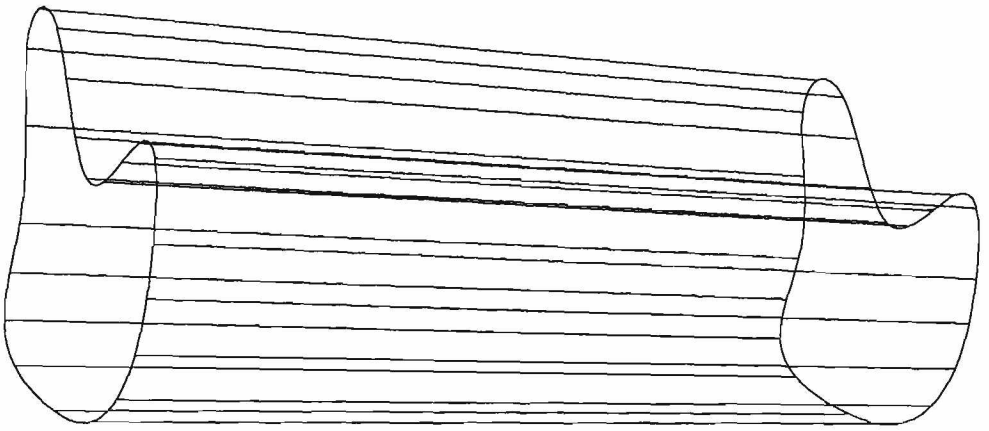

Figure 25: A linear extrusion.

If, in addition, the axis curve of a translational sweep is linear, we call the sweep a **linear extrusion** (Figure 25).

7.2 Rotation Minimizing Offset and Sweep

7.2.1 Curves Composed of Straight Lines and Arcs

A vector which is fixed relative to a rotation minimizing frame is also fixed relative to the Frenet frame, on any planar portion of a curve over which the Frenet frame is defined (see Section 6.4). On such an interval, an offset of a curve $\boldsymbol{\gamma}(t)$ with respect to a rotation minimizing frame is also an offset with respect to the Frenet frame.

Such an offset has the form:

$$\begin{aligned} \mathbf{o}(t) &= \boldsymbol{\gamma}(t) + \left[\begin{array}{ccc} \mathbf{M}_q^N(t) & \mathbf{M}_q^B(t) & \mathbf{T}(t) \end{array} \right] \left(\begin{array}{c} x \\ y \\ z \end{array} \right) \\ &= \boldsymbol{\gamma}(t) + \left[\begin{array}{ccc} \mathbf{N}(t) & \mathbf{B}(t) & \mathbf{T}(t) \end{array} \right] \left(\begin{array}{c} x' \\ y' \\ z \end{array} \right). \end{aligned} \tag{11}$$

The functions $\mathbf{T}$, $\mathbf{N}$, and $\mathbf{B}$, for a rational B-spline curve, are generally not expressible as rational B-splines. Therefore the offset of a rational B-spline

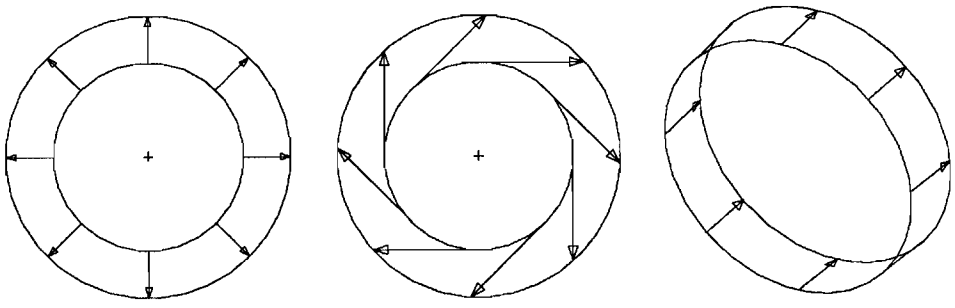

Figure 26: The offset of a circle with respect to the Frenet frame always yields a circle. Shown are offsets in the anti-normal, tangent, and binormal directions.

curve $\boldsymbol{\gamma}$ as given by equation (11) is not, in general, a rational B-spline.

Curves that are straight lines or arcs of a circle are an exception. The offset of a straight line relative to a rotation minimizing frame is simply a translation of the original line (since the functions $\mathbf{M}_q^N$, $\mathbf{M}_q^B$, and $\mathbf{T}$ are constant on the line). Figure 26 illustrates that the offset of a circle with respect to the Frenet frame (and hence with respect to a rotation minimizing frame) is always a circle. We note that an offset in the direction (or anti-direction) of the curve normal results in a simple scaling of the circle about its center. An offset in the direction of the tangent results in both a scaling and a rotation about the circle's center. And an offset in the direction of the binormal results in a simple translation of the circle.

The offset with respect to a rotation minimizing frame of a curve composed entirely of straight lines and arcs (unit tangent continuous) is therefore itself a curve composed entirely of straight lines and arcs, and so is representable exactly as a rational B-spline. (We are ignoring self-intersections and other degeneracies in the result.)

For a sweep along an axis curve $\mathbf{a}(u)$ relative to a rotation minimizing frame, the $\boldsymbol{\alpha}_i(u)$ for the sweep are all offsets of $\mathbf{a}(u)$ relative to the rotation minimizing frame. If $\mathbf{a}(u)$ is composed entirely of straight lines and arcs (unit tangent continuous), the $\boldsymbol{\alpha}_i(u)$ are all exactly representable as rational B-splines. In this case the jth polynomial piece of each $\boldsymbol{\alpha}_i(u)$ is simply an affine transformation of the jth polynomial piece of $\mathbf{a}(u)$. It is therefore

possible to represent corresponding polynomial pieces of the $\boldsymbol{\alpha}_i(u)$ over the same B-spline basis and set of homogeneous coordinates (see Appendix A). We can then represent all the $\boldsymbol{\alpha}_i(u)$ over the same B-spline basis and set of homogeneous coordinates. Such a sweep is therefore exactly representable by a tensor product B-spline of the form given by equation (6) of Section 3.4.

7.2.2 Specialized Control Point Offset Operator

Below we give a specialized Euclidian control point offset operator for use in approximating sweeps and offsets with respect to rotation minimizing frames. This control point offset operator replaces those given in the algorithms of Sections 4.1 and 4.2. The resulting algorithms have the important property that offsets of, or sweeps along, piecewise Bézier curves composed entirely of straight lines and arcs (unit tangent continuous) are exact. Offsets of, or sweeps along, other curves yield approximate results that can be made arbitrarily close to the correct result with refinement of the curve to be offset, or the axis of the sweep (see Appendix B).

For a fixed but arbitrarily chosen i, let:

- $\boldsymbol{\gamma}(t)$ be a NURBS curve,
- $t^* = t_i^*$ be the ith parametric node of the knot vector for $\boldsymbol{\gamma}(t)$,
- $\mathbf{E} = \mathbf{E}_i$ be the Euclidian projection of the ith control point of $\boldsymbol{\gamma}(t)$,
- $\mathbf{V} = (x, y, z)$ be the offset vector,
- $\mathbf{M}_q^N(t)$ and $\mathbf{M}_q^B(t)$ be rotation minimizing normal and rotation minimizing binormal functions for $\boldsymbol{\gamma}(t)$,
- $\mathbf{N}(t)$, $\mathbf{B}(t)$, and $\mathbf{T}(t)$ define the Frenet frame, as before, and
- $\kappa(t)$ be the curvature function for $\boldsymbol{\gamma}(t)$.

The Euclidian control point offset of $\mathbf{E}$ with respect to curve $\boldsymbol{\gamma}(t)$, offset vector $\mathbf{V}$, and frame $\left[\begin{array}{ccc} \mathbf{M}_q^N(t) & \mathbf{M}_q^B(t) & \mathbf{T}(t) \end{array} \right]$ is given by:

$$\mathbf{E}' = \mathbf{E} + \lambda(v_n\hat{\mathbf{N}} + v_t\hat{\mathbf{T}}) + v_b\mathbf{B}(t^*),$$

where:

$$\begin{aligned}
\mathbf{V}' &= x\mathbf{M}_q^N(t^*) + y\mathbf{M}_q^B(t^*) + z\mathbf{T}(t^*), \\
v_n &= \mathbf{V}' \cdot \mathbf{N}(t^*), \\
v_b &= \mathbf{V}' \cdot \mathbf{B}(t^*), \\
v_t &= \mathbf{V}' \cdot \mathbf{T}(t^*) = z, \\
\mathbf{S} &= \boldsymbol{\gamma}(t^*), \\
\mathbf{H} &= \mathbf{E} - \mathbf{S}, \\
\tilde{\mathbf{N}} &= \mathbf{N}(t^*) - \kappa(t^*)\mathbf{H}, \\
\lambda &= \|\tilde{\mathbf{N}}\|, \\
\hat{\mathbf{N}} &= \frac{1}{\lambda}\tilde{\mathbf{N}}, \text{ and} \\
\hat{\mathbf{T}} &= \hat{\mathbf{N}} \times \mathbf{B}(t^*).
\end{aligned}$$

The above control point offset operator is a generalization and simplification of the one developed by Coquillart [6] (with the elimination of Newton iteration and extension to 3-D cross-section curves).

See Appendix D for a derivation and further discussion of this operator.

7.3 Profile Products

In this section we discuss a technique in which a cross-section is swept along a linear axis, while a specified profile curve provides both scaling and orientation information. This notion is a simple extension of the formulation of **spherical products** given in [1].

Without loss of generality, consider the sweep to proceed along the positive z-axis. The cross-section is permitted to be an arbitrary 3-D curve, but it is most intuitive to think of the cross-section as lying in the xy plane. The profile curve is a general 3-D curve with extent in the positive z direction.

The variable scaling of the cross-section is given by the distance from the z-axis to the profile curve as the cross-section sweeps along the z-axis. The

local orientation frame revolves with a vector from the z-axis to the profile curve in a manner discussed below (see Figure 27).

More formally, for cross-section curve $\mathbf{c}(v) = (c_1(v), c_2(v), c_3(v))$ and profile curve $\mathbf{p}(u) = (p_1(u), p_2(u), p_3(u))$, the profile product sweep is defined as:

$$\begin{aligned} \boldsymbol{\sigma}(u,v) &= \mathbf{a}(u) + \left[\begin{array}{ccc} \bar{\mathbf{X}}(u) & \bar{\mathbf{Y}}(u) & \mathbf{Z}(u) \end{array} \right] \mathbf{c}(v) \\ &= \mathbf{a}(u) + \mathcal{F}(u) \begin{pmatrix} g(u)c_1(v) \\ g(u)c_2(v) \\ c_3(v) \end{pmatrix}, \end{aligned} \tag{12}$$

where:

$$\begin{aligned} \mathbf{a}(u) &= (0, 0, p_3(u)), \\ \bar{\mathbf{X}}(u) &= \begin{pmatrix} p_1(u) \\ p_2(u) \\ 0 \end{pmatrix}, \\ \bar{\mathbf{Y}}(u) &= \begin{pmatrix} -p_2(u) \\ p_1(u) \\ 0 \end{pmatrix}, \\ \mathbf{Z}(u) &= \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \\ \mathcal{F}(u) &= \left[\begin{array}{ccc} \mathbf{X}(u) & \mathbf{Y}(u) & \mathbf{Z}(u) \end{array} \right], \\ g(u) &= \|\bar{\mathbf{X}}(u)\|, \\ \mathbf{X}(u) &= \bar{\mathbf{X}}(u)/g(u), \text{ and} \\ \mathbf{Y}(u) &= \bar{\mathbf{Y}}(u)/g(u). \end{aligned}$$

The frame $\mathcal{F}(u)$ is formed so as to be orthonormal. Note that the scaling derived from the profile curve is applied only to the x and y components of the cross-section curve. This results in a scaling about the z-axis as we sweep the cross-section along it.

The helical gear surface of Figure 14 is an example of this type of sweep where the profile is a B-spline approximation to a helix about the z-axis.

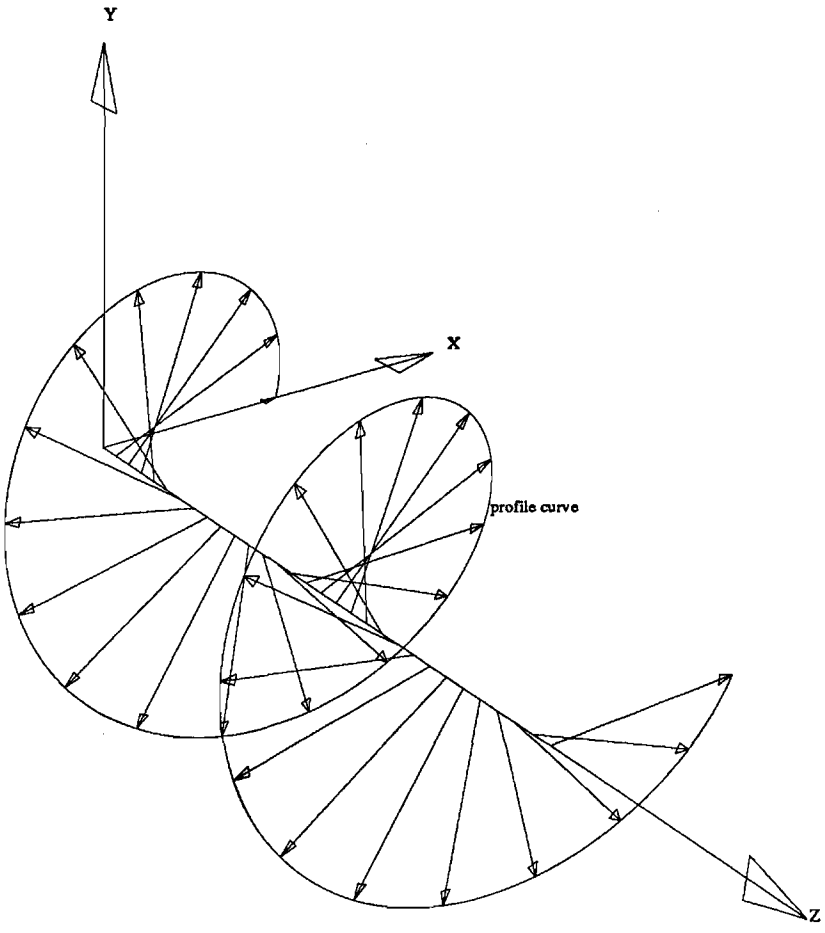

Figure 27: The local orientation frame on the z-axis is allowed to rotate with a vector from the z-axis to the profile curve (and perpendicular to the z-axis). The variable scaling of the cross-section is determined by the length of this vector. The profile curve shown is a helix centered about the z-axis.

For a rational B-spline cross-section curve:

$$\mathbf{c}(v) = \frac{\sum_i h_i^c \mathbf{C}_i B_i^c(v)}{\sum_i h_i^c B_i^c(v)},$$

with:

$$\mathbf{C}_i = (x_i^c, y_i^c, z_i^c),$$

the $\boldsymbol{\alpha}_i(u)$ for the sweep of equation (12) are given by:

$$\begin{aligned}
\boldsymbol{\alpha}_i(u) &= \mathbf{a}(u) + \left[\begin{array}{ccc} \bar{\mathbf{X}}(u) & \bar{\mathbf{Y}}(u) & \mathbf{Z}(u) \end{array}\right] \mathbf{C}_i \\
&= (0,0,p_3(u)) + \begin{bmatrix} p_1(u) & -p_2(u) & 0 \\ p_2(u) & p_1(u) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_i^c \\ y_i^c \\ z_i^c \end{pmatrix} \\
&= (0,0,z_i^c) + \begin{bmatrix} x_i^c & -y_i^c & 0 \\ y_i^c & x_i^c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} p_1(u) \\ p_2(u) \\ p_3(u) \end{pmatrix}.
\end{aligned}$$

This last expression can been seen as a rotation and scaling of the profile curve about the z-axis plus a translation along the z-axis. Thus the $\boldsymbol{\alpha}_i(u)$ are simple affine transformations of $\mathbf{p}(u)$ and hence, if $\mathbf{p}(u)$ is a rational B-spline, $\boldsymbol{\sigma}(u,v)$ is exactly representable by a tensor product B-spline of the form given in equation (6) of Section 3.4.

For

$$\mathbf{p}(u) = \frac{\sum_j h_j^p \mathbf{P}_j B_j^p(u)}{\sum_j h_j^p B_j^p(u)},$$

with:

$$\mathbf{P}_j = (x_j^p, y_j^p, z_j^p),$$

we have:

$$\boldsymbol{\alpha}_i(u) = \frac{\sum_j h_j^p \mathbf{P}'_{ij} B_j^p(u)}{\sum_j h_j^p B_j^p(u)},$$

where:

$$\mathbf{P}'_{ij} = (x_i^c x_j^p - y_i^c y_j^p, y_i^c x_j^p + x_i^c y_j^p, z_i^c + z_j^p).$$

The sweep surface $\boldsymbol{\sigma}(u, v)$ is then given by the tensor product B-spline:

$$\boldsymbol{\sigma}(u, v) = \frac{\sum_i \sum_j h_j^p h_i^c \mathbf{P}'_{ij} B_j^p(u) B_i^c(v)}{\sum_i \sum_j h_j^p h_i^c B_j^p(u) B_i^c(v)}.$$

If the cross-section is a circle in the xy plane then the result is a surface of revolution about the z-axis, and hence we can think of this technique as a generalized surface of revolution.

This approach is readily generalized to sweeps along any linear axis in space. Coquillart [7] formulates a generalization for curved axes.

8 Examples

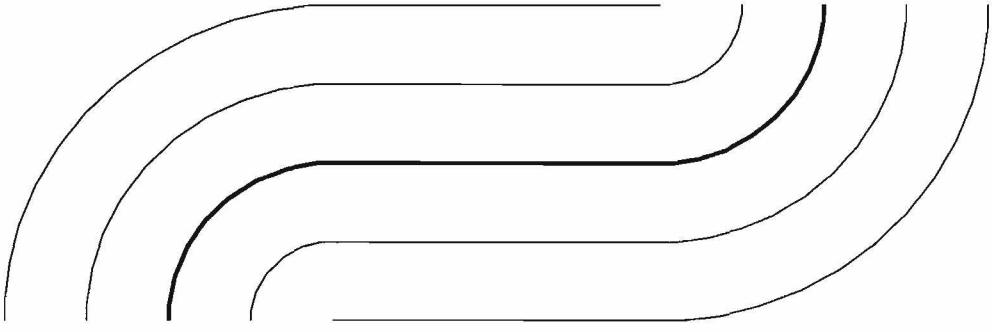

Figure 28: Offsets relative to a rotation minimizing frame along a curve composed of straight lines and arcs. The results are exact.

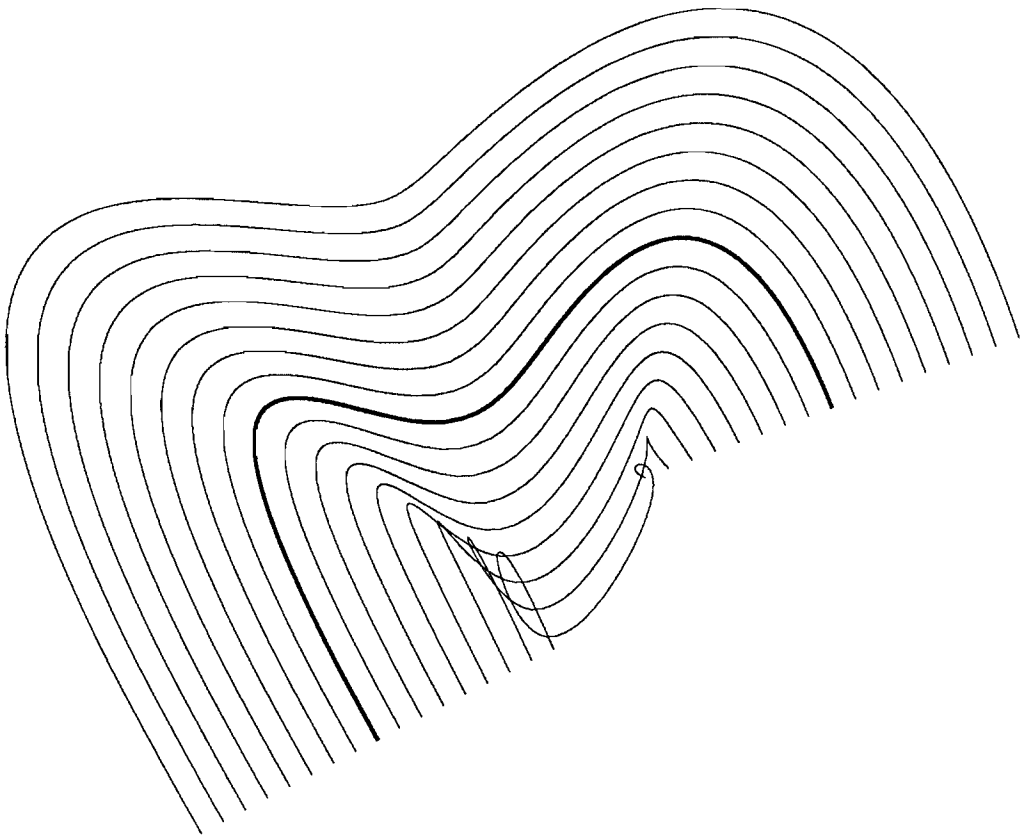

Figure 29: Approximate curve offsets without refinement of the curve being offset (in boldface).

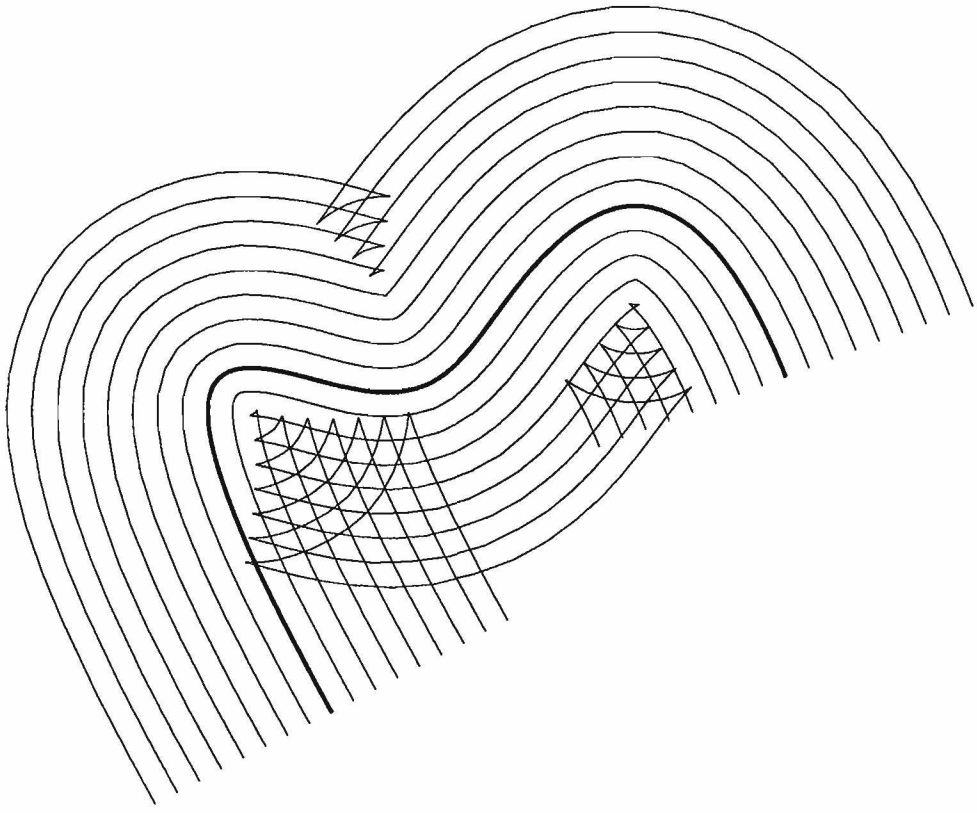

Figure 30: Approximate curve offsets *with* refinement of the curve being offset (in boldface).

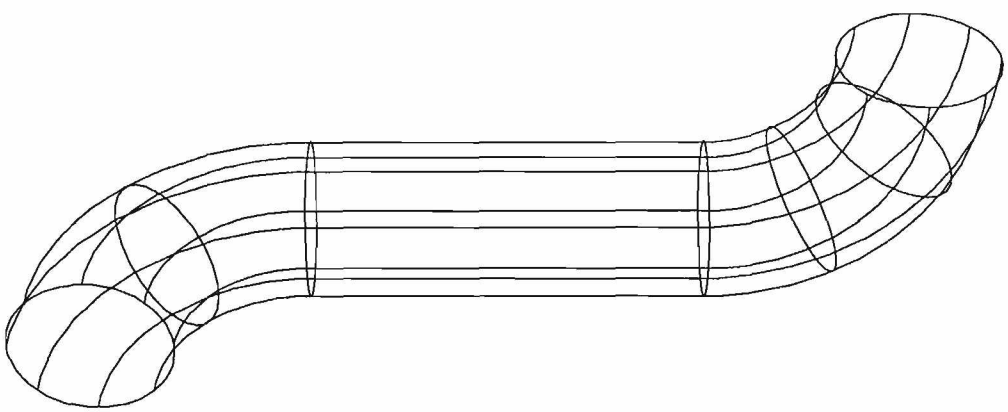

Figure 31: Sweep with respect to a rotation minimizing frame along an axis curve composed of straight lines and arcs. The result is exact.

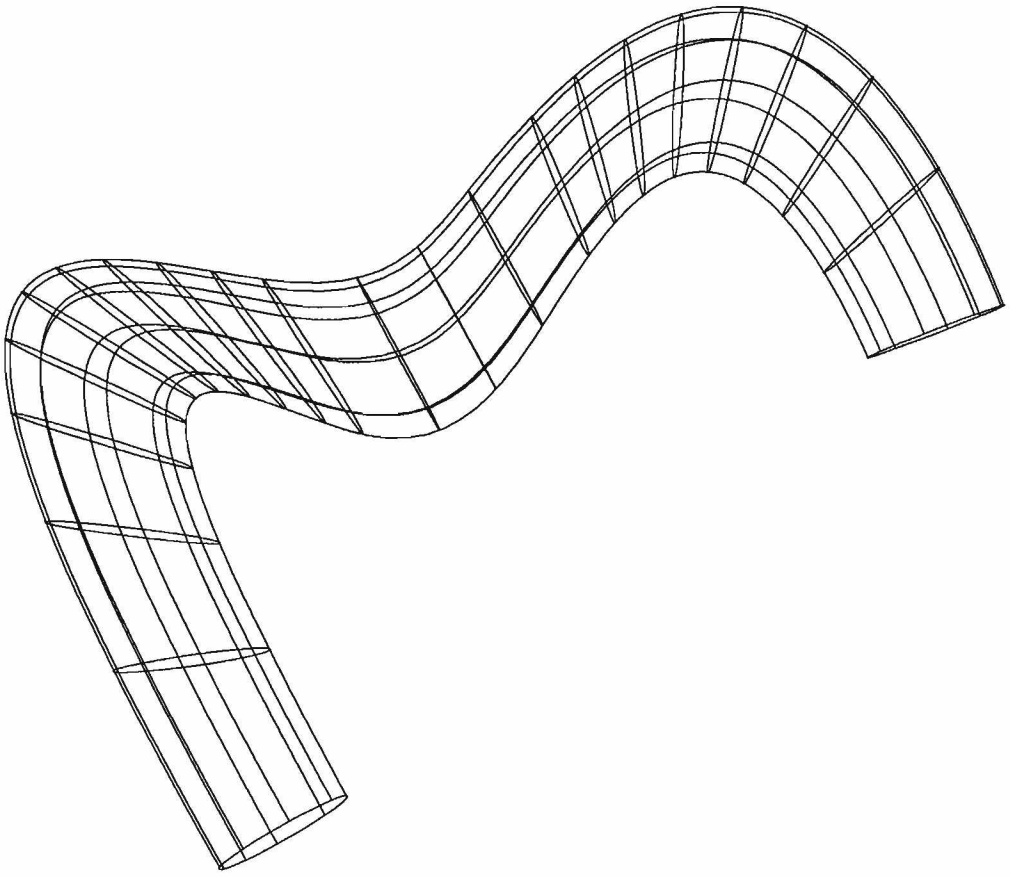

Figure 32: Sweep along a general planar axis curve with *no* refinement of the axis curve.

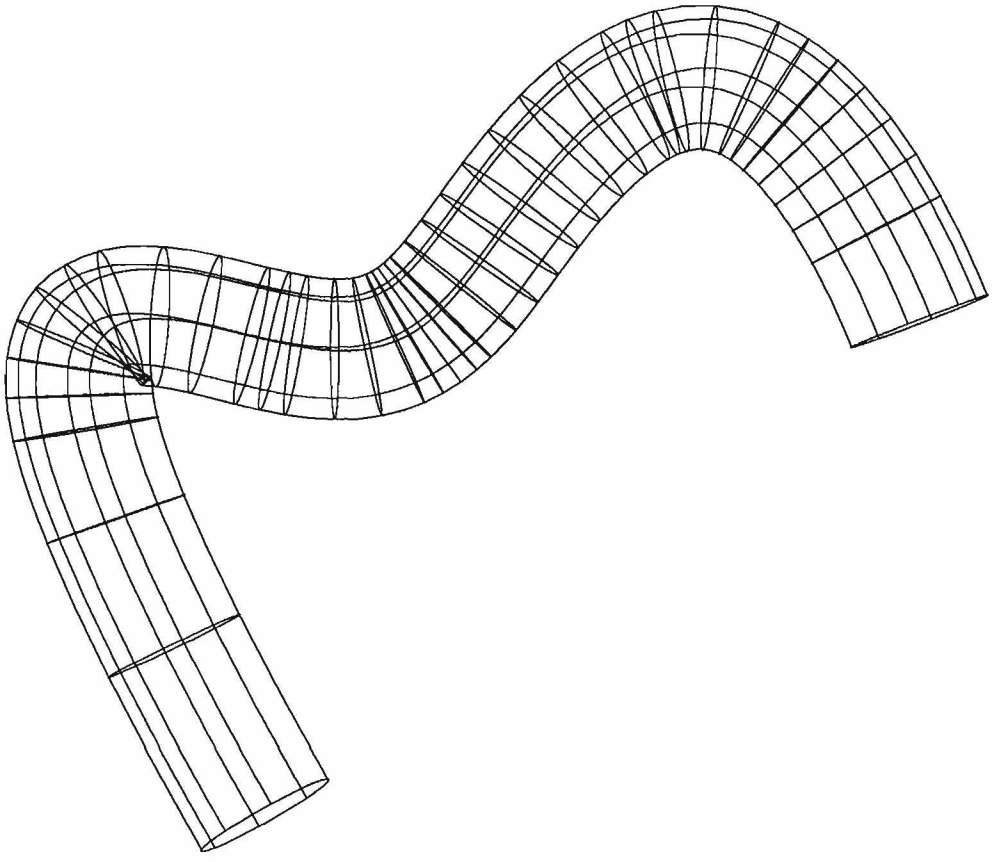

Figure 33: Sweep along the axis curve of Figure 32 after refinement was applied to the axis curve.

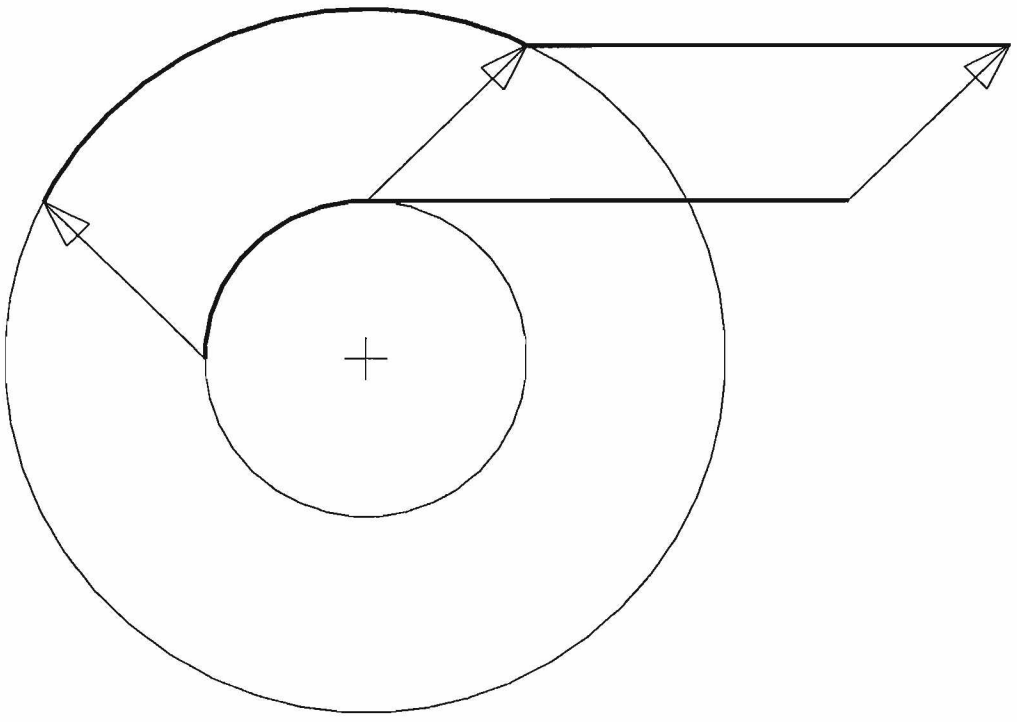

Figure 34: Offsets by a vector with a component in the tangent direction of the curve may yield unintuitive results. The curve being offset is tangent continuous; the result is not.

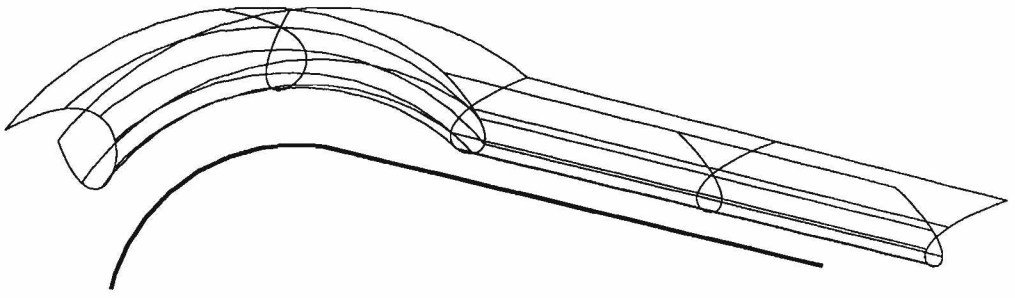

Figure 35: Similarly, sweeps using cross-section curves that do not lie in the xy plane may yield unintuitive results. The axis curve is the same tangent continuous curve used in Figure 34. The sweep is relative to a rotation minimizing frame on the axis curve.

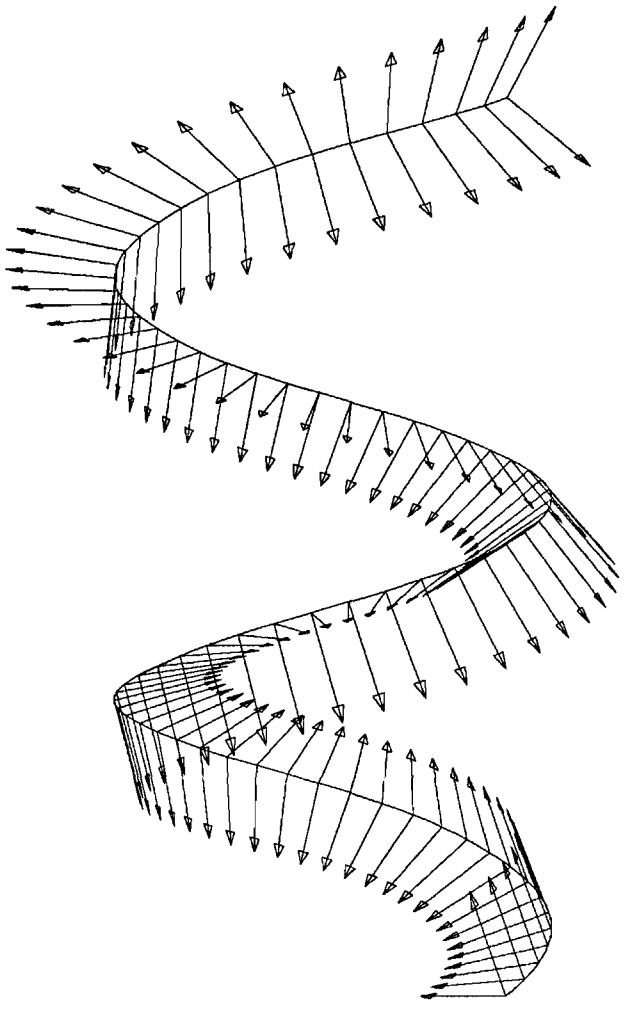

Figure 36: Rotation minimizing frames do not always yield intuitive sweeps and offsets. Shown here is a rotation minimizing frame along a helix. The Frenet frame is a better choice in this case.

A Affine Transformations of B-splines

In this section we prove that a rational B-spline can undergo an affine transformation by applying that transformation to the Euclidian projections of its control points.

Let the affine transformation be denoted as: $\mathcal{A} = \mathcal{L} + \mathbf{v}$ for some linear transformation $\mathcal{L}$ and some vector $\mathbf{v}$.

Then:

$$\begin{aligned}
\mathcal{A}\left[\frac{\sum_i h_i \mathbf{E}_i B_i(t)}{\sum_i h_i B_i(t)}\right] &= \mathcal{L}\left[\frac{\sum_i h_i \mathbf{E}_i B_i(t)}{\sum_i h_i B_i(t)}\right] + \mathbf{v} \\
&= \frac{\sum_i h_i \mathcal{L}[\mathbf{E}_i] B_i(t)}{\sum_i h_i B_i(t)} + \frac{\mathbf{v}\sum_i h_i B_i(t)}{\sum_i h_i B_i(t)}, \qquad \text{since } \mathcal{L} \text{ is linear.} \\
&= \frac{\sum_i h_i \mathcal{L}[\mathbf{E}_i] B_i(t)}{\sum_i h_i B_i(t)} + \frac{\sum_i h_i \mathbf{v} B_i(t)}{\sum_i h_i B_i(t)} \\
&= \frac{\sum_i h_i (\mathcal{L}[\mathbf{E}_i] + \mathbf{v}) B_i(t)}{\sum_i h_i B_i(t)} \\
&= \frac{\sum_i h_i \mathcal{A}[\mathbf{E}_i] B_i(t)}{\sum_i h_i B_i(t)}.
\end{aligned}$$

B Convergence of a Sequence of Offset or Sweep Approximations

In this section we prove the convergence of a sequence of offset approximations that use the algorithms of Sections 4.1 and 7.2 (the simple offset and rotation minimizing offset algorithms). This also proves, by (8), the convergence of a sequence of sweep approximations that use the algorithms of Sections 4.2 and 7.2. Finally we show the convergence of a sequence of sweep approximations that use the algorithms of Section 5 (sweeps with deforming cross-section).

B.1 Notation

$\vec{t_n} = (t_{n,0}, t_{n,1}, \cdots)$ is the nth knot vector in a sequence of knot vectors, each a refinement of the previous one.

$\|\vec{t}_n\| = \max(t_{n,1} - t_{n,0},\ t_{n,2} - t_{n,1}, \cdots)$ is the **mesh norm** of $\vec{t}_n$.

$t^*_{n,j}$ is the jth parametric node of $\vec{t}_n$ for a fixed order k.

B.2 Basic Lemmas

Let $\gamma(t)$ be a NURBS curve of order k and knot vector $\vec{t} = (t_0, t_1, \cdots)$. We consider the curve $\gamma_n(t) = \gamma(t)$ as defined over a sequence of knot vectors $\vec{t}_n$, each a refinement of the previous one, such that,

$$\vec{t}_0 = \vec{t} \text{ and } \|\vec{t}_n\| \to 0 \text{ as } n \to \infty.$$

The projective control points of $\gamma_n(t)$ are,

$$\mathbf{P}_{n,j} = (\mathbf{G}_{n,j}, h_{n,j}),$$

where $\mathbf{G}_{n,j} = h_{n,j}\mathbf{E}_{n,j}$, with $\mathbf{E}_{n,j}$ the Euclidian projection of $\mathbf{P}_{n,j}$.

We have,

$$\gamma_n(t) = \frac{\sum_j \mathbf{G}_{n,j} B_{n,j}(t)}{\sum_j h_{n,j} B_{n,j}(t)} = \frac{\mathbf{g}(t)}{h(t)},$$

where we assume $h(t)$ is bounded away from zero.

For a fixed but arbitrarily chosen value of t find $j_n : t \in [t_{n,j_n}, t_{n,j_n+1})$. Then,

$$\gamma_n(t) = \frac{\sum_{j=j_n-k+1}^{j_n} \mathbf{G}_{n,j} B_{n,j}(t)}{\sum_{j=j_n-k+1}^{j_n} h_{n,j} B_{n,j}(t)}.$$

Lemma 1

$$\boxed{t^*_{n,j} \to t \text{ as } n \to \infty \text{ for } j = j_n - k + 1, \cdots, j_n.}$$

Proof:

Since $\|\vec{t}_n\| \to 0$ as $n \to \infty$, $t_{n,j} \to t$ as $n \to \infty$ for $j = j_n - k + 2, \cdots, j_n + k - 1$. The lemma follows trivially.

Lemma 2

$$\boxed{h_{n,j} \to h(t) \text{ as } n \to \infty \text{ for } j = j_n - k + 1, \cdots, j_n.}$$

Proof:

We have,

$$\|h_{n,j} - h(t)\| \leq \|h_{n,j} - h(t^*_{n,j})\| + \|h(t^*_{n,j}) - h(t)\| \text{ for } j = j_n - k + 1, \cdots, j_n.$$

a) From lemma 1, and the assumed continuity of $h(t)$, we know that,

$$\|h(t^*_{n,j}) - h(t)\| \to 0 \text{ as } n \to \infty \text{ for } j = j_n - k + 1, \cdots, j_n.$$

b) For $\mathbf{L}_n$ the linear operator of [5],

$$\|\mathbf{L}_n h(t) - h(t)\| \to 0 \text{ for all } t \text{ as } n \to \infty.$$

In particular,

$$\|\mathbf{L}_n h(t^*_{n,j}) - h(t^*_{n,j})\| \to 0 \text{ as } n \to \infty, \text{ for } j = j_n - k + 1, \cdots, j_n.$$

But $\mathbf{L}_n h(t^*_{n,j}) = h_{n,j}$ so,

$$\|h_{n,j} - h(t^*_{n,j})\| \to 0 \text{ as } n \to \infty, \text{ for } j = j_n - k + 1, \cdots, j_n.$$

Parts a) and b) together imply the lemma.

Lemma 3

$$\boxed{\mathbf{E}_{n,j} \to \gamma(t) \text{ as } n \to \infty \text{ for } j = j_n - k + 1, \cdots, j_n.}$$

Proof:

$$\|\mathbf{G}_{n,j} - \mathbf{g}(t)\| \leq \|\mathbf{G}_{n,j} - \mathbf{g}(t^*_{n,j})\| + \|\mathbf{g}(t^*_{n,j}) - \mathbf{g}(t)\|$$

for $j = j_n - k + 1, \cdots, j_n$.

a) From lemma 1, and the assumed continuity of $\mathbf{g}(t)$, we know that,

$$\|\mathbf{g}(t^*_{n,j}) - \mathbf{g}(t)\| \to \infty \text{ as } n \to 0 \text{ for } j = j_n - k + 1, \cdots, j_n.$$

b) For $\mathbf{L}_n$ the linear operator of [5],

$$\|\mathbf{L}_n\mathbf{g}(t) - \mathbf{g}(t)\| \to \infty \text{ for all } t \text{ as } n \to 0.$$

In particular,

$$\|\mathbf{L}_n\mathbf{g}(t^*_{n,j}) - \mathbf{g}(t^*_{n,j})\| \to 0 \text{ as } n \to \infty \text{ for } j = j_n - k + 1, \cdots, j_n.$$

But $\mathbf{L}_n\mathbf{G}(t^*_{n,j}) = \mathbf{G}_{n,j}$ so,

$$\|\mathbf{G}_{n,j} - \mathbf{g}(t^*_{n,j})\| \to 0 \text{ as } n \to \infty \text{ for } j = j_n - k + 1, \cdots, j_n.$$

Parts a) and b) together imply,

$$\mathbf{G}_{n,j} \to \mathbf{g}(t) \text{ as } n \to \infty \text{ for } j = j_n - k + 1, \cdots, j_n.$$

But by lemma 2,

$$\mathbf{E}_{n,j} = \mathbf{G}_{n,j}/h_{n,j} \to \mathbf{g}(t)/h(t) = \boldsymbol{\gamma}(t) \text{ as } n \to \infty,$$

for $j = j_n - k + 1, \cdots, j_n$.

B.3 Offset Algorithm 1

Using $\boldsymbol{\gamma}_n$ in the curve offset algorithm of Section 4.1 yields,

$$\hat{\mathbf{o}}_n(t) = \frac{\sum_j h_{n,j}\mathbf{E}'_{n,j}B_{n,j}(t)}{\sum_j h_{n,j}B_{n,j}(t)},$$

where $\mathbf{E}'_{n,j} = \mathbf{E}_{n,j} + \boldsymbol{\mathcal{F}}(t^*_{n,j})\mathbf{V}$.

We will show that,

$$\hat{\mathbf{o}}_n(t) \rightarrow \mathbf{o}(t) = \boldsymbol{\gamma}(t) + \boldsymbol{\mathcal{F}}(t)\mathbf{V} \text{ as } n \rightarrow \infty.$$

For a fixed but arbitrarily chosen value of t find $j_n : t \in [t_{n,j_n}, t_{n,j_n+1})$.

From lemmas 1, 2, and 3 we have,

$$\begin{aligned}
\hat{\mathbf{o}}_n(t) &= \frac{\sum_{j=j_n-k+1}^{j_n} h_{n,j}(\mathbf{E}_{n,j} + \boldsymbol{\mathcal{F}}(t^*_{n,j})\mathbf{V})B_{n,j}(t)}{\sum_{j=j_n-k+1}^{j_n} h_{n,j}B_{n,j}(t)} \\
&\rightarrow \frac{\sum_{j=j_n-k+1}^{j_n} h(t)(\boldsymbol{\gamma}(t) + \boldsymbol{\mathcal{F}}(t)\mathbf{V})B_{n,j}(t)}{\sum_{j=j_n-k+1}^{j_n} h(t)B_{n,j}(t)} \\
&= \boldsymbol{\gamma}(t) + \boldsymbol{\mathcal{F}}(t)\mathbf{V} = \mathbf{o}(t),
\end{aligned}$$

as $n \rightarrow \infty$.

B.4 Offset Algorithm 2

Using $\boldsymbol{\gamma}_n$ in the rotation minimizing offset algorithm of Section 7.2 yields,

$$\hat{\mathbf{o}}_n(t) = \frac{\sum_j h_{n,j}\mathbf{E}'_{n,j}B_{n,j}(t)}{\sum_j h_{n,j}B_{n,j}(t)},$$

where,

$$\begin{aligned}
\mathbf{E}'_{n,j} = \mathbf{E}_{n,j} \;+\; & \lambda_{n,j}[\,(\mathbf{V}'_{n,j} \cdot \mathbf{N}(t^*_{n,j}))\hat{\mathbf{N}}_{n,j} + (\mathbf{V}'_{n,j} \cdot \mathbf{T}(t^*_{n,j}))\hat{\mathbf{T}}_{n,j}\,] \;+ \\
& (\mathbf{V}'_{n,j} \cdot \mathbf{B}(t^*_{n,j}))\mathbf{B}_{n,j}(t^*_{n,j}).
\end{aligned}$$

We will show that,

$$\hat{\mathbf{o}}_n(t) \rightarrow \boldsymbol{\gamma}(t) + \boldsymbol{\mathcal{F}}(t)\mathbf{V} = \mathbf{o}(t) \text{ as } n \rightarrow \infty.$$

For a fixed but arbitrarily chosen value of t find $j_n : t \in [t_{n,j_n}, t_{n,j_n+1})$.
We assume that the functions, $\boldsymbol{\gamma}$, $\mathbf{T}$, $\mathbf{N}$, $\mathbf{B}$, $\mathbf{M}_q^N$, and $\mathbf{M}_q^B$, are continuous in a neighborhood of t, and that $\kappa(t)$ is bounded in this neighborhood. Then, from lemmas 1 and 3, the following holds for $j = j_n - k + 1, \cdots, j_n$:

$$\begin{aligned}
\mathbf{E}_{n,j} &\rightarrow \boldsymbol{\gamma}(t), \\
\mathbf{S}_{n,j} &\rightarrow \boldsymbol{\gamma}(t), \\
\mathbf{N}(t^*_{n,j}) &\rightarrow \mathbf{N}(t), \\
\mathbf{B}(t^*_{n,j}) &\rightarrow \mathbf{B}(t), \\
\mathbf{T}(t^*_{n,j}) &\rightarrow \mathbf{T}(t), \\
\mathbf{M}_q^N(t^*_{n,j}) &\rightarrow \mathbf{M}_q^N(t), \\
\mathbf{M}_q^B(t^*_{n,j}) &\rightarrow \mathbf{M}_q^B(t), \text{ and} \\
\mathbf{V}'_{n,j} &\rightarrow \mathbf{V}'(t) = \left[\begin{array}{ccc} \mathbf{M}_q^N(t) & \mathbf{M}_q^B(t) & \mathbf{T}(t) \end{array} \right] \mathbf{V} = \boldsymbol{\mathcal{F}}(t)\mathbf{V}
\end{aligned}$$

as $n \rightarrow \infty$.

This implies,

$$\begin{aligned}
\mathbf{E}_{n,j} - \mathbf{S}_{n,j} &\rightarrow \mathbf{0}, \\
\tilde{\mathbf{N}}_{n,j} &\rightarrow \mathbf{N}(t), \\
\lambda_{n,j} &\rightarrow 1, \\
\hat{\mathbf{N}}_{n,j} &\rightarrow \mathbf{N}(t), \text{ and} \\
\hat{\mathbf{T}}_{n,j} &\rightarrow \mathbf{T}(t)
\end{aligned}$$

as $n \rightarrow \infty$.

And so,

$$\begin{aligned}
\mathbf{E}'_{n,j} &\rightarrow \boldsymbol{\gamma}(t) + (\mathbf{V}'(t) \cdot \mathbf{N}(t))\mathbf{N}(t) + (\mathbf{V}'(t) \cdot \mathbf{B}(t))\mathbf{B}(t) + (\mathbf{V}'(t) \cdot \mathbf{T}(t))\mathbf{T}(t) \\
&= \boldsymbol{\gamma}(t) + \mathbf{V}'(t) = \boldsymbol{\gamma}(t) + \boldsymbol{\mathcal{F}}(t)\mathbf{V}
\end{aligned}$$

as $n \rightarrow \infty$.

This now reduces to the case for the previous algorithm.

B.5 Sweep with Deforming Cross-Section

A sweep surface with constant cross-section is given by,

$$\boldsymbol{\sigma}(u,v) = \frac{\sum_i h_i^c \boldsymbol{\alpha}_i(u) B_i^c(v)}{\sum_i h_i^c B_i^c(v)},$$

with $\boldsymbol{\alpha}_i(u) = \mathbf{a}(u) + \boldsymbol{\mathcal{F}}(u)\mathbf{C}_i$.

For the sweep extension, we consider only deformations of the cross-section that can be expressed as deformations of its control polygon. Each projective point of the cross-section's polygon has the form,

$$h_i^c(u)(\mathbf{C}_i(u), 1).$$

Substituting we get,

$$\boldsymbol{\sigma}(u,v) = \frac{\sum_i h_i^c(u) \boldsymbol{\alpha}_i(u) B_i^c(v)}{\sum_i h_i^c(u) B_i^c(v)},$$

where $\boldsymbol{\alpha}_i(u) = \mathbf{a}(u) + \boldsymbol{\mathcal{F}}(u)\mathbf{C}_i(u)$.

Let the axis curve, $\mathbf{a}(u)$, be a NURBS curve of order k and knot vector $\vec{u}$. We consider the curve $\mathbf{a}_n(u) = \mathbf{a}(u)$ as defined over a sequence of knot vectors $\vec{u}_n$, each a refinement of the previous one, such that,

$$\vec{u}_0 = \vec{u} \text{ and } \|\vec{u}_n\| \to 0 \text{ as } n \to \infty.$$

The projective control points of $\mathbf{a}_n(u)$ are,

$$\mathbf{P}_{n,j} = (\mathbf{G}_{n,j}, h^a_{n,j}),$$

where $\mathbf{G}_{n,j} = h^a_{n,j}\mathbf{A}_{n,j}$, with $\mathbf{A}_{n,j}$ the Euclidian projection of $\mathbf{P}_{n,j}$.

We have,

$$\mathbf{a}_n(u) = \frac{\sum_j \mathbf{G}_{n,j} B^a_{n,j}(u)}{\sum_j h^a_{n,j} B^a_{n,j}(u)} = \frac{\sum_j \mathbf{G}_{n,j} B^a_{n,j}(u)}{h^a(u)}.$$

Using $\mathbf{a}_n(u)$ in the sweep with deforming cross-section algorithm of Section 5 yields,

$$\hat{\boldsymbol{\sigma}}_n(u,v) = \frac{\sum_i \sum_j h^a_{n,j} h^c_i(u^*_{n,j}) \mathbf{A}'_{n,i,j} B^a_{n,j}(u) B^c_i(v)}{\sum_i \sum_j h^a_{n,j} h^c_i(u^*_{n,j}) B^a_{n,j}(u) B^c_i(v)},$$

where $\mathbf{A}'_{n,i,j} = \mathbf{A}_{n,j} + \boldsymbol{\mathcal{F}}(u^*_{n,j})\mathbf{C}_i(u^*_{n,j})$.

We will show that,

$$\hat{\boldsymbol{\sigma}}_n(u,v) \to \boldsymbol{\sigma}(u,v) \text{ as } n \to \infty.$$

For a fixed but arbitrarily chosen value of u find $j_n : u \in [u_{n,j_n}, u_{n,j_n+1})$. Then,

$$\mathbf{a}_n(u) = \frac{\sum_{j=j_n-k+1}^{j_n} \mathbf{G}_{n,j} B^a_{n,j}(u)}{\sum_{j=j_n-k+1}^{j_n} h^a_{n,j} B^a_{n,j}(u)}.$$

Let,

$$\begin{aligned} \mathbf{q}_i(u) &= h^a(u)h^c_i(u)(\mathbf{a}(u) + \boldsymbol{\mathcal{F}}(u)\mathbf{C}_i(u)) = h^a(u)h^c_i(u)\boldsymbol{\alpha}_i(u)\,, \text{ and} \\ \hat{\mathbf{q}}_{n,i,j} &= h^a_{n,j}h^c_i(u^*_{n,j})[\mathbf{A}_{n,j} + \boldsymbol{\mathcal{F}}(u^*_{n,j})\mathbf{C}_i(u^*_{n,j})] \text{ for } j = j_n - k + 1, \cdots, j_n. \end{aligned}$$

From lemmas 1, 2 , and 3,

$$\hat{\mathbf{q}}_{n,i,j} \to \mathbf{q}_i(u) \text{ as } n \to \infty \text{ for } j = j_n - k + 1, \cdots, j_n.$$

For a given ϵ choose N such that for $j = j_n - k + 1, \cdots, j_n$ and all i we have $\|\hat{\mathbf{q}}_{n,i,j} - \mathbf{q}_i(u)\| \leq \epsilon$ whenever $n \geq N$. Then

$$\begin{aligned} \|\sum_{j=j_n-k+1}^{j_n} \hat{\mathbf{q}}_{n,i,j} B^a_{n,j}(u) \; - \; & \sum_{j=j_n-k+1}^{j_n} \mathbf{q}_i(u) B^a_{n,j}(u)\| \\ & \leq \sum_{j=j_n-k+1}^{j_n} \epsilon B^a_{n,j}(u) = \epsilon, \end{aligned}$$

whenever $n \geq N$, since $\sum_{j=j_n-k+1}^{j_n} B^a_{n,j} \equiv 1$.

Therefore,

$$\boxed{\sum_{j=j_n-k+1}^{j_n} \hat{\mathbf{q}}_{n,i,j} B^a_{n,j} \to \sum_{j=j_n-k+1}^{j_n} \mathbf{q}_i(u) B^a_{n,j}(u) = \mathbf{q}_i(u) \text{ as } n \to \infty.}$$

Let,

$$\begin{aligned} r_i(u) &= h^a(u)h^c_i(u)\,, \text{ and} \\ \hat{r}_{n,i,j} &= h^a_{n,j}h^c_i(u^*_{n,j}) \text{ for } j = j_n - k + 1, \cdots, j_n. \end{aligned}$$

From lemmas 1 and 2,

$$\hat{r}_{n,i,j} \to r_i \text{ as } n \to \infty \text{ for } j = j_n - k + 1, \cdots, j_n.$$

For a given ϵ choose N such that for $j = j_n - k + 1, \cdots, j_n$ and all i we have $|\hat{r}_{n,i,j} - r_i| \leq \epsilon$ whenever $n \geq N$. Then

$$\begin{aligned} |\textstyle\sum_{j=j_n-k+1}^{j_n} \hat{r}_{n,i,j} B^a_{n,j}(u) \; - \; \sum_{j=j_n-k+1}^{j_n} r_i(u) B^a_{n,j}(u)| \\ \leq \; \textstyle\sum_{j=j_n-k+1}^{j_n} \epsilon B^a_{n,j}(u) = \epsilon, \end{aligned}$$

whenever $n \geq N$.

Therefore,

$$\boxed{\textstyle\sum_{j=j_n-k+1}^{j_n} \hat{r}_{n,i,j} B^a_{n,j} \to \sum_{j=j_n-k+1}^{j_n} r_i(u) B^a_{n,j}(u) = r_i(u) \text{ as } n \to \infty.}$$

For the fixed value of u,

$$\begin{aligned} \hat{\boldsymbol{\sigma}}_n(u,v) = & \\ & \frac{\sum_i \sum_{j=j_n-k+1}^{j_n} h^a_{n,j} h^c_i(u^*_{n,j}) [\mathbf{A}_{n,j} + \boldsymbol{\mathcal{F}}(u^*_{n,j}) \mathbf{C}_i(u^*_{n,j})] B^a_{n,j}(u) B^c_i(v)}{\sum_i \sum_{j=j_n-k+1}^{j_n} h^a_{n,j} h^c_i(u^*_{n,j}) B^a_{n,j}(u) B^c_i(v)} \\ = & \frac{\sum_i \sum_{j=j_n-k+1}^{j_n} \hat{\mathbf{q}}_{n,i,j} B^a_{n,j}(u) B^c_i(v)}{\sum_i \sum_{j=j_n-k+1}^{j_n} \hat{r}_{n,i,j} B^a_{n,j}(u) B^c_i(v)} \\ \to & \frac{\sum_i \mathbf{q}_i B^c_i(v)}{\sum_i r_i B^c_i(v)} \\ = & \frac{\sum_i h^a(u) h^c_i(u) \boldsymbol{\alpha}_i(u) B^c_i(v)}{\sum_i h^a(u) h^c_i(u) B^c_i(v)} = \frac{\sum_i h^c_i(u) \boldsymbol{\alpha}_i(u) B^c_i(v)}{\sum_i h^c_i(u) B^c_i(v)} \\ = & \; \boldsymbol{\sigma}(u,v) \end{aligned}$$

as $n \to \infty$.

C Cubic Polynomials and Inflection Points

If an inflection point (or cusp) exists on a cubic polynomial, then the polynomial must be planar.

Proof:

Let $\mathbf{p}(t)$ be a cubic polynomial.

Assume there exists a point t_0 such that the curvature goes to zero, i.e.:

$$\frac{\|\mathbf{p}'(t_0) \times \mathbf{p}''(t_0)\|}{\|\mathbf{p}'(t_0)\|^3} = 0.$$

Then we have:

$$\mathbf{p}'(t_0) \times \mathbf{p}''(t_0) = \mathbf{0}, \qquad \text{where } \mathbf{0} \text{ is the zero vector.}$$

Note the case where the curvature is undefined is included in this case.

We must have:

$$\begin{aligned} \mathbf{p}'(t_0) &= \mathbf{0} \qquad \text{or} \\ \mathbf{p}''(t_0) &= k\mathbf{p}'(t_0), \end{aligned}$$

for some scalar k.

We can write $\mathbf{p}(t)$ as the Taylor expansion about the point t_0:

$$\mathbf{p}(t) = \mathbf{p}(t_0) + (t - t_0)\mathbf{p}'(t_0) + 1/2(t - t_0)^2\mathbf{p}''(t_0) + 1/6(t - t_0)^3\mathbf{p}'''(t_0).$$

If $\mathbf{p}'(t_0) = \mathbf{0}$ then the $\mathbf{p}'$ term in the Taylor expansion contributes nothing. In this case $\mathbf{p}(t)$ has been written as a point plus a linear combination of two vectors, and hence $\mathbf{p}(t)$ is a planar curve.

If $\mathbf{p}''(t_0) = k\mathbf{p}'(t_0)$ then we can write:

$$\mathbf{p}(t) = \mathbf{p}(t_0) + [(t - t_0) + 1/2k(t - t_0)^2]\mathbf{p}'(t_0) + 1/6(t - t_0)^3\mathbf{p}'''(t_0).$$

So $\mathbf{p}(t)$ is again written as a point plus a linear combination of two vectors, and hence $\mathbf{p}(t)$ is planar.

A similar result holds in the case of $\mathbf{p}(t)$ a rational cubic polynomial.

D Control Point Offset Operator for use by Rotation Minimizing Offsets and Sweeps

In this section we develop a specialized control point offset operator for use in approximating offsets and sweeps with respect to a rotation minimizing frame.

It is assumed throughout this section that curves being offset or curves used as axes of sweeps are unit tangent continuous.

D.0.1 Offsets of a Circle

We consider the offset of $\gamma(t)$, a rational B-spline representation for a circle. We offset $\gamma(t)$ by the vector $\mathbf{V} = (x, y, z)$ relative to the rotation minimizing frame $\left[\begin{array}{ccc} \mathbf{M}_q^N(t) & \mathbf{M}_q^B(t) & \mathbf{T}(t) \end{array} \right]$ (for some choice of an initial rotation minimizing normal, $\mathbf{q}$). The resulting offset curve is an affine transformation of $\gamma(t)$ (scaled, rotated and translated; see Section 7.2.1). We can effect this transformation by applying it individually to each of the Euclidian projections of the control points for γ (see Appendix A).

We develop a Euclidian control point offset operator analogous to the one used in the simple offset algorithm of Section 4.1. The result of applying this offset operator to the Euclidian projection of a control point of circle γ is that the point undergoes the proper affine transformation. This control point offset operator can then be used in place of equation (9) in the offset algorithm of Section 4.1 to yield exact results for the offset of circle $\gamma(t)$ relative to a rotation minimizing frame. In computing the offset of a control point we use only *local* geometry from the curve γ. In this way we can then extend the method to approximations for offsets of more general curves.

Figure 37 illustrates the geometry for the control point offset operator applied to the Euclidian projection $\mathbf{E}$ of a control point of $\gamma(t)$. $\mathbf{S} = \gamma(t')$ is a point lying on circle $\gamma(t)$ that we associate with $\mathbf{E}$. For the time being we let t' be an arbitrary value in the domain of $\gamma(t)$. We use the local geometry of $\gamma(t)$ at $\mathbf{S}$ to determine how to offset $\mathbf{E}$. $\mathbf{V}' = \left[\begin{array}{ccc} \mathbf{M}_q^N(t') & \mathbf{M}_q^B(t') & \mathbf{T}(t') \end{array} \right] \mathbf{V}$ is the offset vector $\mathbf{V}$ interpreted relative to the local frame at $\mathbf{S}$. $\mathbf{S}'$ is the point on the *offset* curve associated with $\mathbf{S}$. $\mathbf{O}$ is the center of circle γ and r is its radius. Only the components of $\mathbf{V}'$ in the direction of the curve's normal and tangent at $\mathbf{S}$ are shown. The component in the binormal direction is

considered later.

Given $\mathbf{E}$ and $\mathbf{S}$, we compute $\mathbf{E}'$ which is the Euclidian control point offset of $\mathbf{E}$. Since the resulting offset circle is simply a scaled and rotated version of the original circle, $\triangle\mathbf{OE'S'}$ is a scaled and rotated version of $\triangle\mathbf{OES}$. We introduce points $\hat{\mathbf{S}}$ and $\hat{\mathbf{E}}$ as shown in Figure 37 and prove that $\triangle\mathbf{E\hat{E}E'} \cong \triangle\mathbf{S\hat{S}S'}$.

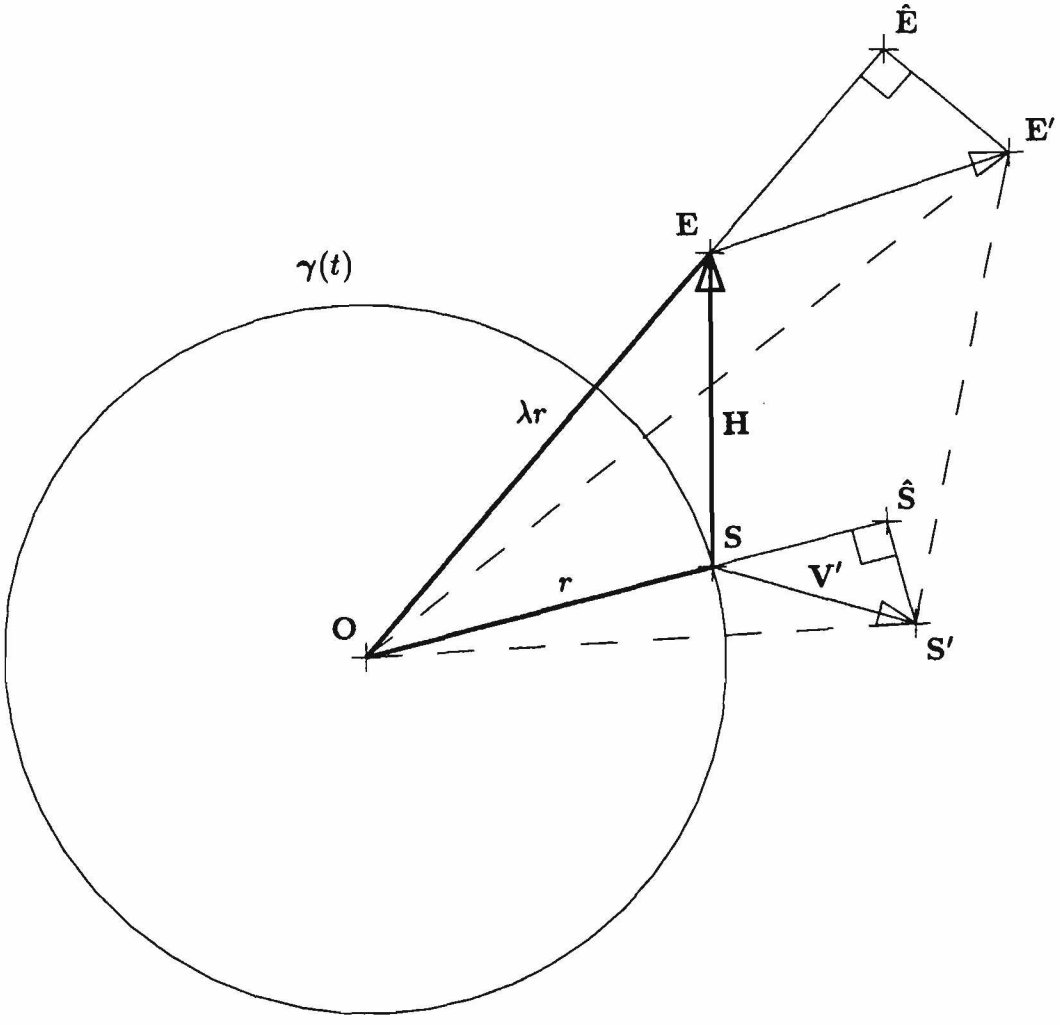

Figure 37: Derivation of the control point offset operator.

We are given that $\triangle \mathbf{OE'S'} \cong \triangle \mathbf{OES}$.

Let $\lambda = \frac{\|\overline{\mathbf{OE}}\|}{\|\overline{\mathbf{OS}}\|} \Rightarrow \|\overline{\mathbf{OE}}\| = \lambda\|\overline{\mathbf{OS}}\|$ and $\|\overline{\mathbf{OE'}}\| = \lambda\|\overline{\mathbf{OS'}}\|$.

Also $\angle \mathbf{EOS} = \angle \mathbf{E'OS'} \Rightarrow \angle \mathbf{EOE'} = \angle \mathbf{SOS'}$.

By side-angle-side similarity, we have:

$$\triangle \mathbf{OEE'} \cong \triangle \mathbf{OSS'} \text{ with } \|\overline{\mathbf{EE'}}\| = \lambda\|\overline{\mathbf{SS'}}\|.$$

$\angle \hat{\mathbf{E}}\mathbf{EE'}$ is external to $\triangle \mathbf{OEE'}$ and $\angle \hat{\mathbf{S}}\mathbf{SS'}$ is external to $\triangle \mathbf{OSS'} \Rightarrow \angle \hat{\mathbf{E}}\mathbf{EE'} = \angle \hat{\mathbf{S}}\mathbf{SS'}$ and since $\angle \mathbf{E'}\hat{\mathbf{E}}\mathbf{E} = \angle \mathbf{S'}\hat{\mathbf{S}}\mathbf{S} = \pi/2$ we have:

$$\triangle \mathbf{E}\hat{\mathbf{E}}\mathbf{E'} \cong \triangle \mathbf{S}\hat{\mathbf{S}}\mathbf{S'} \text{ with } \|\overline{\mathbf{EE'}}\| = \lambda\|\overline{\mathbf{SS'}}\|. \tag{13}$$

We now derive the Euclidian control point offset formula by constructing a rotated "Frenet frame" basis for $\triangle \mathbf{E}\hat{\mathbf{E}}\mathbf{E'}$ (see Figure 38).

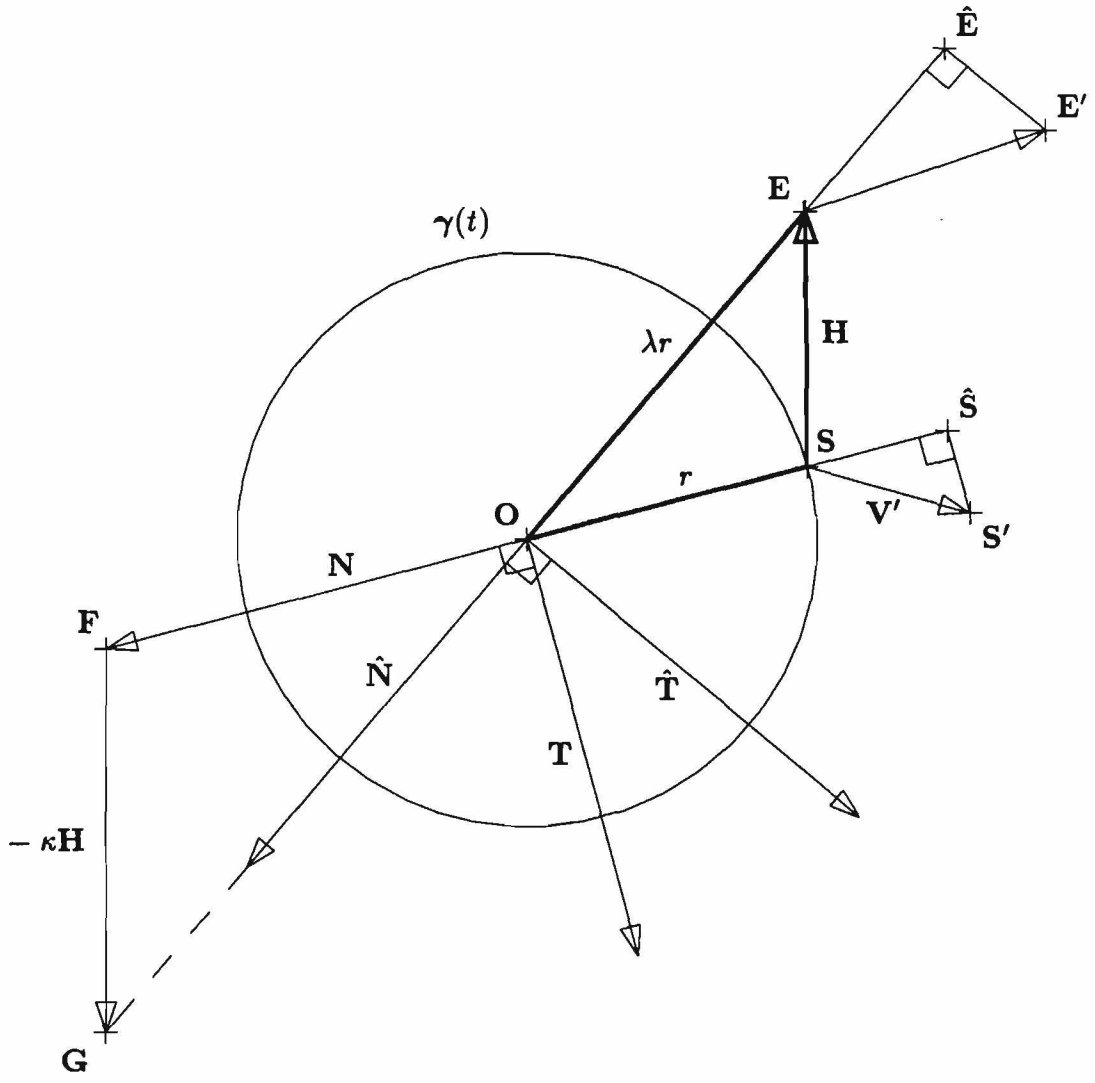

Figure 38: Derivation of the control point offset operator (continued).

Let $\mathbf{T}$, $\mathbf{N}$ and $\mathbf{B}$ be the unit tangent, normal, and binormal of $\boldsymbol{\gamma}$ at $\mathbf{S}$.

Let point $\mathbf{F} = \mathbf{O} + \mathbf{N}$.

Let point $\mathbf{G}$ be on the line containing $\overline{\mathbf{OE}}$ as shown such that $\overline{\mathbf{FG}}$ is parallel to $\overline{\mathbf{SE}}$.

$\triangle\mathbf{OFG} \cong \triangle\mathbf{OSE}$ since $\angle\mathbf{FOG} = \angle\mathbf{SOE}$ and $\angle\mathbf{OFG} = \angle\mathbf{OSE}$.

We then have:

$$\frac{\|\overline{\mathbf{FG}}\|}{\|\overline{\mathbf{SE}}\|} = \frac{\|\overline{\mathbf{OF}}\|}{\|\overline{\mathbf{OS}}\|} = \frac{1}{r} \Rightarrow \overline{\mathbf{FG}} = -\frac{1}{r}\overline{\mathbf{SE}},$$

and

$$\frac{\|\overline{\mathbf{OE}}\|}{\|\overline{\mathbf{OS}}\|} = \frac{\|\overline{\mathbf{OG}}\|}{\|\overline{\mathbf{OF}}\|} = \|\overline{\mathbf{OG}}\| = \lambda.$$

Using the additional fact that $\|\overline{\mathbf{EE'}}\| = \lambda\|\overline{\mathbf{SS'}}\|$ from equation (13) we can now compute the offset control point $\mathbf{E}'$ as follows:

Let v_n and v_t be the components of $\mathbf{V}'$ in the direction of the normal and tangent at $\mathbf{S}$ respectively.

$$\begin{aligned}
\mathbf{H} &= \mathbf{E} - \mathbf{S}, \\
\tilde{\mathbf{N}} &= \overline{\mathbf{OG}} = \mathbf{N} - \kappa\mathbf{H}, \text{ where } \kappa = \tfrac{1}{r}, \\
\lambda &= \|\tilde{\mathbf{N}}\|, \\
\hat{\mathbf{N}} &= \frac{1}{\lambda}\tilde{\mathbf{N}}, \\
\hat{\mathbf{T}} &= \hat{\mathbf{N}} \times \mathbf{B}, \text{ and} \\
\mathbf{E}' &= \mathbf{E} + \lambda(v_n\hat{\mathbf{N}} + v_t\hat{\mathbf{T}}).
\end{aligned} \tag{14}$$

If there is a component v_b of $\mathbf{V}'$ in the direction of the binormal at $\mathbf{S}$, this component simply gives rise to a translation resulting in:

$$\mathbf{E}' = \mathbf{E} + \lambda(v_n\hat{\mathbf{N}} + v_t\hat{\mathbf{T}}) + v_b\mathbf{B}.$$

In summary, this control point offset operator computes $\mathbf{E}'$ as an affine transformation of $\mathbf{E}$ using only local geometric information from the circle

γ. The affine transformation is the one that transforms the original circle γ into the offset circle. We can offset γ correctly by applying this control point offset operation to each of the Euclidian projections of the control points for γ.

D.0.2 Generalization to Arbitrary Curves

We note that the above derivation does not depend on any particular choice of **S** to associate with **E**. Any point on the circle would do. However for general curves, where the geometry is not so regular, we should choose **S** at some point on the curve where the local geometry is influenced by the control point **E** (i.e., where there is an association between **E** and the curve's geometry at **S**). We use the curve value at the parametric node, t^*, associated with **E** (in an analogous manner to the algorithm of Section 4.1).

For control point offset calculations for a general curve, we approximate the curve locally as a circle at the point **S**. We choose the osculating circle to the curve at this point for this purpose [12]. In practice, this means only that the $\kappa = \frac{1}{r}$ of equation (14) now becomes the curvature at the point **S**.

A full statement of the Euclidian control point offset algorithm is given in Section 7.2.2.

D.0.3 What About the Use of the Frenet Frame?

We might expect to have trouble with $\mathbf{N}(t)$, $\mathbf{B}(t)$, and $\kappa(t)$ in the Euclidian control point offset formulation developed here since they are not always well defined on a general curve.

However, if we assume that a unit tangent is everywhere defined, then if **N** is undefined, κ is zero.

With $\kappa = 0$, we have $\hat{\mathbf{N}} = \mathbf{N}$, $\hat{\mathbf{T}} = \mathbf{T}$, and $\lambda = 1$, so:

$$\begin{aligned}
\mathbf{E}' &= \mathbf{E} + \lambda(v_n\hat{\mathbf{N}} + v_t\hat{\mathbf{T}}) + v_b\mathbf{B} \\
&= \mathbf{E} + v_n\mathbf{N} + v_t\mathbf{T} + v_b\mathbf{B} \\
&= \mathbf{E} + (\mathbf{V}' \cdot \mathbf{N})\mathbf{N} + (\mathbf{V}' \cdot \mathbf{T})\mathbf{T} + (\mathbf{V}' \cdot \mathbf{B})\mathbf{B} \\
&= \mathbf{E} + \mathbf{V}'.
\end{aligned}$$

What if we have a discontinuity of the normal at $\mathbf{S} = \boldsymbol{\gamma}(t^*)$? Which of the two normals defined (in the limit) at $\mathbf{S}$ do we use?

If we exclude the case of cusps, in practice we need not make the choice. For then, in order to have a discontinuity of the normal, we must have a knot of multiplicity *at least* order $-$ 2 at t^*. If the knot is of multiplicity order $-$ 2, then it can not be a node (except in the limiting sense that it is a knot of multiplicity order $-$ 1).

If we have a knot of multiplicity order $-$ 1 then we have interpolation (i.e., $\mathbf{E} = \mathbf{S}$ with $\mathbf{H} = \mathbf{E} - \mathbf{S}$ the zero vector) and the formula again becomes:

$$\mathbf{E}' = \mathbf{E} + \mathbf{V}'.$$

D.0.4 Sweep and Offset Algorithms

The Euclidian control point offset operator developed here can be used in place of the control point offset operator given in the simple algorithm of Section 4.1. The resulting curve offset routine is exact for offsets of a circle.

For a curve having zero curvature (i.e., a straight line) the control point offset operator reduces to:

$$\mathbf{E}' = \mathbf{E} + \mathbf{V}',$$

(see Section D.0.3) and hence the resulting offset curve is the appropriate translation of the original straight line.

Now consider the case where a circular arc is embedded as a rational polynomial piece of a curve. Let $\mathbf{I}$ be the parametric interval that spans the embedded arc. When the curve is offset, the embedded arc is offset correctly provided that the parametric nodes associated with each control point actually blended on $\mathbf{I}$ all lie on $\mathbf{I}$. If this is the case, then all points $\mathbf{S}$ associated with these control points will lie on the arc. This insures that the correct local samples of the geometry of the arc are used in computing the offsets of the control points blended on this interval.

The same conditions hold for properly offsetting embedded linear pieces of a curve. All sample points $\mathbf{S}$ used to offset the control points blended over the linear interval, must lie on the embedded linear piece.

These conditions hold for the case of piecewise Bézier curves containing only straight lines and arcs, constructed so as to be unit tangent continuous. So these curves are offset exactly.

For offsets of arbitrary curves, this algorithm produces approximations that can be made arbitrarily close to the exact offset by using refinement of the curve being offset as a preprocess step (see Appendix B).

The Euclidian control point offset operator developed in this section can be used in place of the offset operator given in the simple sweep algorithm of Section 4.2. The resulting sweep algorithm has analogous properties to the offset algorithm. Sweeps with piecewise Bézier axis curves composed entirely of straight lines and arcs, constructed so as to be unit tangent continuous, give exact results. All other axis curves yield approximate results that can be made arbitrarily accurate with refinement of the axis curve as a preprocess step (see Section 4.2).

References

[1] Alan H. Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1), January 1981.

[2] Elizabeth Susan Cobb. *Design of Sculptured Surfaces Using the B-spline Representation.* PhD thesis, University of Utah, Salt Lake City, Utah, June 1984.

[3] E. Cohen, T. Lyche, and R. F. Riesenfeld. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 14(2):87–111, October 1980. Also Tech. Report No. UUCS-79-117, UUCS, October 1979.

[4] E. Cohen, T. Lyche, and L. L. Schumaker. Algorithms for degree-raising of splines. *ACM Transactions on Graphics*, 4(3):171–181, July 1985.

[5] Elaine Cohen and L. L. Schumaker. Rates of convergence of control polygons. *Computer Aided Geometric Design*, 2:229–235, September 1985.

[6] Sabine Coquillart. Computing offsets of b-spline curves. *Computer-Aided Design*, 19(6), July/August 1987.

[7] Sabine Coquillart. A control-point-based sweeping technique. *IEEE Computer Graphics and Applications*, 7(11), November 1987.

[8] C. de Boor. *A Practical Guide to Splines*, volume 27 of *Applied Mathematical Sciences.* Springer-Verlag, 1978.

[9] Carl de Boor. Splines as linear combinations of B-splines. A survey. In G. G. Lorentz, C. K. Chui, and L. L. Schumaker, editors, *Approximation Theory II*, pages 1–47. Academic Press, New York, 1976.

[10] Brian Donahue. Modelling complex objects with generalized sweeps. Master's thesis, University of Utah, Salt Lake City, Utah, 1985.

[11] R. T. Farouki. Exact offset procedures for simple solids. *Computer Aided Geometric Design*, 2:257–279, 1985.

[12] I. D. Faux and M. J. Pratt. *Computational Geometry for Design and Manufacture.* Ellis Horwood, 1985.

[13] W. J. Gordon and R. F. Riesenfeld. B-spline curves and surfaces. In Robert E. Barnhill and Richard F. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 95–126. Academic Press, New York, 1974.

[14] Fopke Klok. Two moving coordinate frames for sweeping along a 3d trajectory. *Computer Aided Geometric Design*, 3:217–229, 1986.

[15] Timothy I. Mueller. Refinement estimation of b-spline curves and surfaces. Technical Report UUCS-88-030, University of Utah, Salt Lake City, Utah, 1988.

[16] Uri Shani and Dana H. Ballard. Splines as embeddings for generalized cylinders. *Computer Vision, Graphics, and Image Processing*, 27(2):129–156, August 1984.